FINAL TECHNICAL REPORT

ON

A STUDY OF THE PRODUCTION PLANNING AND SCHEDULING PROBLEM IN THE APPAREL MANUFACTURING INDUSTRY PHASE II

(#DLA900-91-C-1481)

submitted to

Defense Logistics Agency
Manufacturing Engineering Branch
Cameron Station-DLA-PRM
Alexandria, Virginia 22304-6100
Attn: Mr. Dan Gearing

by

Dr. Chin-Sheng Chen
Industrial & Systems Engineering Department
Florida International University
University Park, ECS 416
Miami, Florida 33199
Tel: (305) 348-3753

March 31, 1995

REPORT DOCUMENTATION PAGE

Form Approved OMB No. [illegible]

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	31 Mar 1995	Final (23 June 92 - 22 Feb 95

4. TITLE AND SUBTITLE
A Study of the Production Planning and Scheduling Problem in the Apparel Manufacturing Industry

5. FUNDING NUMBERS
DLA900-91-C-1481

6. AUTHOR(S)
Dr. Chin-Sheng Chen

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Florida International University
Department of Industrial & Systems Engineering
University Park, ECS 416
Miami, FL 33199

8. PERFORMING ORGANIZATION REPORT NUMBER
571815700

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Electronics Supply Center/DLA
1507 Wilmington Pike
Dayton, OH 45444-5208

Office of Naval Research
ONRRR
Georgia Tech
206 O'Keefe Building
Atlanta, GA 30332

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
none

12a. DISTRIBUTION/AVAILABILITY STATEMENT
DIST "A"

DISTRIBUTION STATEMENT A
Approved for public release
Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)
The project seeks the development of an on-line apparel production planning and scheduling system based on an accurate estimate of time-phased apparel plant capacity. The system implementation consists of two major components: an operator's skill prediction module and a production planning & scheduling module. The first one is developed to predict the skill progress of individual operators with 17 operation-related factors, and to estimate the capacity of an apparel plant over a given planning horizon. The estimate provides a quantitative foundation for generation of accurate production plans and promotes operator's long-term skill growth. Several heuristic algorithms including simulated annealing and neural network are embedded in the scheduling algorithm. The system is coded in C++ and implemented on a PC/386 platform with support of the Paradox database management system.

14. SUBJECT TERMS	15. NUMBER OF PAGES
production planning, scheduling, sehop loading, and learning	157
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

EXECUTIVE SUMMARY

This report summarizes the study of the apparel production planning and scheduling problem and the implementation of a computer integrated apparel production scheduling system. Most of small and medium sized apparel manufacturing firms are engaged in make-to-order production, which is quite different from make-to-stock production in that: 1) no inventory is available to offset demands fluctuation, and 2) the capacity of a manufacturing plant largely relies on product structure and worker's momentary skills. The survival of a make-to-order company thus depends on its ability to quickly respond to an order inquiry and its flexibility in production operations. One focus of this study is an accurate estimate of time-phased plant capacity such that realistic production schedule can be made. Such an estimate becomes difficult to an apparel firm when it faces a sharp production surge or a dramatic garment style change such that a substantial hiring and training is required.

The proposed apparel production scheduling system consists of eight major computer modules: 1) a database and database manager, 2) a graphic user interface, 3) a sewing skill predictor, 4) a plant capacity planner, 5) a master production scheduler, 6) a detailed production planner, 7) a material flow simulator, and 8) a production method evaluator. The apparel database is developed to store required product, production, and manufacturing resource data to support system applications. The database manager is created to operate and maintain the database. The skill predictor estimates time-phased sewing skills of individual workers and the plant capacity. Based on accurate prediction, the production planning and scheduling modules are applied to generate weekly production plans and daily work assignments, which are then further refined by the material flow simulator.

The proposed system is coded in Turbo C++ and runs in the Microsoft DOS/Windows environment on PC/486s. The Paradox database management system is used to implement a relational database and the database manager. Each application module interacts with (retrieve from and contribute to) the same database. Real-life apparel production data were used in the development and validation of the system prototype. A close work relationship was established with several apparel firms which helped to improve the system. In addition to this final technical report, a user's manual and a software requirements specification for the system are prepared and attached to the report.

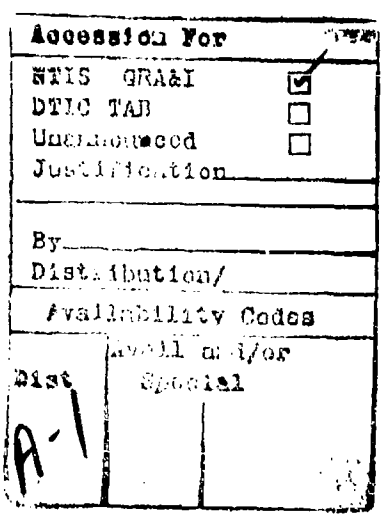


TABLE OF CONTENTS

1. INTRODUCTION

The domestic apparel manufacturing industry has been facing stiff competition from its overseas counterparts and has been fighting an uphill battle. One factor which works toward the advantage of domestic apparel manufacturers is their proximity to the market place and thus can quickly respond to the changes in the market. In fact, quick response and flexible manufacturing have been often considered an important strategy to the survival of this industry. To achieve this goal, many non-traditional manufacturing practices such as unit production, modular manufacturing, and computer integrated manufacturing have been introduced to the industry.

To realize the quick response concept, it is required to have a production planning system which can quickly estimate the manufacturing cost and lead time for a potential order, develop an accurate production plan and schedule, and ensure the availability of the required resources so as to produce and deliver the order promptly. At the decision-making stage of accepting an order, however, it is difficult to accurately generate a detailed production plan, prepare a manning schedule, and estimate manufacturing costs. The level of difficulty increases in a situation where there is a radical change in style, or a large increase in production is required over a short period of time and the company must substantially increase hiring and training activities. This situation commonly occurs when a small apparel manufacturer accepts a contract to produce military apparel. Without accurate production plans and estimates, severe delay in delivery and excessive manufacturing cost may result.

There are several factors contributing to this difficulty. Delays on materials delivery, unexpected absenteeism, high employee turnover, equipment breakdown, and static production planning methodology are common problems. In additions to these, an accurate estimate of the production capacity is another serious challenge. Since the ability to change styles quickly is important to a make-to-order company, it has to rely on general-purpose machines to maintain its flexibility. Thus extensive hiring, training, and retraining activities are required whenever there is a significant change in garment design or in production quantity. It is hard to measure production capacity in this type of environment. The limit of training capacity and the learning effect on production capacity over time are frequently underestimated. These factors usually make a significant contribution to the delay and cost increase. Current production planning and scheduling practices, however, do not fully address the impacts of operations sequence and time-phased training effects on production capacity.

1.1 The Planning and Scheduling Problem

Production planning and scheduling occurs in a wide range of economic activities and has received significant research attention for decades. It is the process of organizing, choosing, and timing resource usage to carry out activities required to produce desired outputs at desired times, while satisfying given constraints on resources, due dates, and other operational limitations.

In a typical definition of a production planning and scheduling system, a job is defined to represent an individual task to be accomplished. Each job consists a number of sub-tasks to

be carried out, which are often called operations or activities. Each operation consumes a certain amount of resources of various types for a duration of time. Resources are typically classified into various types such as machines, tools, workers, and so on. The search for a feasible (or optimal) schedule often involves in solving a large-scale combinatorial problem, which is considered NP-complete [27].

The challenges for solving an NP-complete problem are: 1) computation time required to search the entire feasible solution space and 2) computer memory space required to manage the vast amount of data for the search. Therefore, heuristic approaches are often taken in attempt to identify good solutions within an acceptable computation time, instead of taking an enumerative approach to search for the optimal solution. In the domain of heuristic algorithms, prior research has been focused on dispatching rules. Most heuristic algorithms which base on priority rules are domain-specific and thus only applicable to the problem domain being studied. A review of job shop scheduling and dispatching rules can be found in Graves [15], Panwalker and Iskander [28], Baker [3], and Cheng and Gupta [8]. These algorithms are relatively simple and somewhat limited in their potential application and optimality. Currently the research trend moves toward developing more complicated, dynamic and intelligent heuristic algorithms, using such search concepts as generic algorithms, beam search, tabu search, simulated annealing, neural networks, and AI/expert systems.

1.2 The Apparel Production Planning and Scheduling Problem

Most apparel firms, especially small and medium sized companies, involve in make-to-order and batch production. The production method used in a typical apparel manufacturing firm is job shop in nature. However, the make-to-order apparel production is quite different from job-shop production in many ways. First of all, garment materials travel in bundles of 20 to 30 garment pieces. Secondly the main apparel production activity is sewing (i.e, various assembly operations), although other types of operations such as cutting, pressing, and inspection do exist. Due to frequent style changes and demand fluctuations, sewing operators are often engaged in a different sewing operation at a different proficiency level, making plant capacity a variable and rendering all line-balancing techniques not applicable. The challenges for scheduling an apparel assembly shop are summarized as follow:

1) There exists a complex operation precedence diagram for each garment style.

2) The shop is organized as a "flexible" assembly line, in response to short production runs and the need for handling multiple but non-synchronized products simultaneously.

3) Workstations are manned with multiple skilled workers at varying proficiency levels. Thus the capacities of both workstations and the plant are a variable.

4) Workers may be re-assigned to a different workstation to engage in a different operation at a different skill level.

1.3 Objective and Tasks

The objective of this project is to study, develop and implement a production planning and scheduling system for the apparel manufacturing firms in the assembly-to-order and bundling production environment as described above. The major tasks in this project include the development and implementation of a framework for the production planning and scheduling system, a database system to store apparel design, production, and resources data, production planning and scheduling modules, and a friendly user interface. The six major application modules to be focused in this study are for sewing skill prediction, master production scheduling, detailed production scheduling, plant capacity planning, material flow simulation, and production method evaluation.

In addition to typical scheduling functionalities, the proposed system is intended to take into account the factors which affect production planning and a precise estimate of manufacturing lead time and manufacturing cost. Another focus of this study is the development of an integrated database system which maintains data integrity and supports concurrent planning and scheduling activities. It should also be easily extensible in the future to support and integrate other application functions. The search for a good solution technique for efficiently solving real-life apparel scheduling problem is also an important task in this study.

1.4 Assumptions

Based on the planning horizon, production planing may be considered as: 1) long range planning (2-5 years), 2) medium range planning (1-2 years), or 3) short range planning (3-6 months). A planning horizon shorter than 3 months is usually considered as scheduling, which typically generates a detailed schedule and may consider reactive control. Due to the nature of short production runs in the apparel industry, the study is focused on short-term planning and scheduling activities. Major assumptions and constraints are summarized as follows:

1) It is engaged in assemble-to-order production.

2) Each bundle of garment is routed through any workstation no more than once.

3) Each order comes with a due date which is set by the customer and no penalty is assessed for early delivery.

4) One workstation perforins only one type of sewing operation.

5) Workers may perform different operation types at a different skill level.

6) The sewing skill of an individual worker for an operation type varies, depending on the amount of training and the operation and operator related factors.

7) The efficiency of a workstation thus depends on the efficiency of workers assigned to the

station at that time.

1.5 Structure of the Report

The engineering design of the proposed apparel production planning and scheduling system will be presented in Chapter 2, which details the system framework, the required engineering data, the user interface, and individual engineering functions. The implementation of the system is presented in Chapter 3. Concluding remarks are provided in Chapter 4. A user's manual is included as Appendix A1. The software requirements specification is attached as Appendix A2.

2. SYSTEM DESIGN

Production planning and scheduling is concerned with the effective management of the total flow of goods from the supply of raw materials to the delivery of finished products in right quantities, right quality, right time, and at a reasonable cost [4]. It is a rather complex engineering process and usually takes place as a sequence of decision activities and requires a variety of engineering data. These decisions are hierarchically inter-related. The decision made at a higher level provides additional constraints for lower-level decision-making; and lower level decisions, in turn, provide feedback to validate the upper level decisions [30]. To make an effective decision, various engineering data must be constantly updated in the process and often be applied to project the needs and resource availability in the future. Due to the nature of apparel production (i.e., short production runs, make-to-order, variable sewing skills and plant capacity, and short manufacturing lead time), the planning system must allow quick decision-making and respond to changes efficiently. Momentary optimization of an apparel manufacturing system is not critical, because of the system dynamics. To operate a decision system in such an environment, the use of computer is viewed as a must. It ensures data integrity and speedy computation.

A hierarchical approach is applied to solving the apparel production planning and scheduling problem and to development of the planning and scheduling system. The development procedure followed in this study can be summarized as follows:

1) determine the functions and performance criteria for the system.

2) structure the system and set its system boundary.

3) identify significant components that make up the system.

4) develop and implement each individual system component in line with the overall system.

5) define the I/O and interface requirements for the system components.

6) develop and implement a database and a database manager.

7) integrate and test the system.

Major functions of an apparel production planning and scheduling system include: production method selection, master production scheduling, and detailed production planning. This study is focused on bundle-based apparel production system, which is considered the most popular apparel production method in the industry. An evaluation of various production methods can tell the manager if there exists a better production method. Master production scheduling may involve in capacity planning when demands exceed current plant capacity. In the apparel industry, capacity planning may involve skill prediction, because sewing skills changes, depending on worker assignment and amount of training. Both master production schedule and

detailed production schedule are static schedules. They do not take into account the interactions of production activities on the shop floor, where delays and penalties often occur because of bottlenecks and no resources. A simulation tool can be used to verify and fine tune a static production plan in a given production environment. It could be further applied as a production control tool, when linked to an on-line shop floor control system. These functions frequently interact with one another and must share a set of common product and resource data. A database is therefore essential to ensure the sharing and integrity of apparel life cycle data. The major engineering functions included in the proposed system are highlighted below:

1) graphic user interface (GUI)

   The graphic user interface provides the user a structured menu to operate the system in an interactive and friendly environment.

2) production method evaluation (PME)

   It enables the system to identify the best production method for a given job or order.

3) sewing skill prediction (SSP)

   It enables the system to predict sewing skills of a worker at a given operation at any given time.

4) plant capacity planning (PCP)

   It enables the system to estimate the time-phased plant capacity, reflecting the progression of sewing skills of the workers and incorporating available external resources.

5) master production scheduling (MPS)

   It enables the system to generate a weekly production schedule and to evaluate the feasibility of new order(s) as well as existing ones.

6) detailed production scheduling (DPS)

   It enables the system to detail the master production schedule into worker assignments to jobs and workstations.

7) material flow simulation (MFS)

   It enables the system to validate static production plans in a given production environment. It could also be used as a production control tool, when linked to an on-line shop floor control system.

8) database manager

It enables the system to maintain an integrated apparel production database which support the above planning and scheduling applications. It could also support other enterprise functions.

The proposed planning and scheduling system can be viewed as a decision support system, which has a database, a user interface, and a set of computation-based procedures. The computational procedures in this system are the above engineering applications, each making one specific production decision with some user input. Because of the database support, the system may engage in a vast amount of information processing activities but requires only minimum input data directly from the user during the decision process. All required apparel design, production, and resource data have been stored in the database and maintained by the system. The system architecture is detailed in section 2.1, which is followed by the database design in section 2.2. The remaining sections in this chapter present the planning and scheduling procedures developed for the above engineering applications.

2.1 System Architecture

A framework for the proposed system is presented in Figure 1. The system contains the following modules: a database manager, a graphic user interface, a production method evaluator, a sewing skill predictor, a training scheduler, a plant capacity planner, a schedule feasibility evaluator, a master production scheduler, a detailed production scheduler, and a material flow simulator. Among them, the feasibility evaluation and the training capacity planning modules are not a focus of this study.

The graphic user interface (GUI) provides a system-wide structured menu for the user to efficiently operate the system. Through the GUI design, the user may receive orders, update resources, add product and production knowledge, browse production schedules, and run an application module. Before other module is applied, the production method evaluator (PME) should be activated to evaluate available production methods for each new order and assess the appropriateness of using the bundle production method, the most commonly used production method in the apparel industry.

The master production scheduler (MPS) is the central piece of the system, which generates and maintains a weekly production plan for a fixed number of weeks, and ensures that the resources are adequate. It sets a weekly production level for each job order and the manning level by operation type. The planning is done according to the current plant capacity and the projection of time-phased plant capacity in the future. The challenge is that both capacities are a variable, depending on the path of worker assignment and thus their skills at these operations to be decided. The sewing skill predictor (SSP) is therefore needed, which predicts time-phased worker's skill at a projected worker assignment history. The plant capacity planner is applied to select external resources to satisfy additional capacity need, when in-house and regular plant capacity can not ensure meeting due dates, which may be caused by accepting new orders or

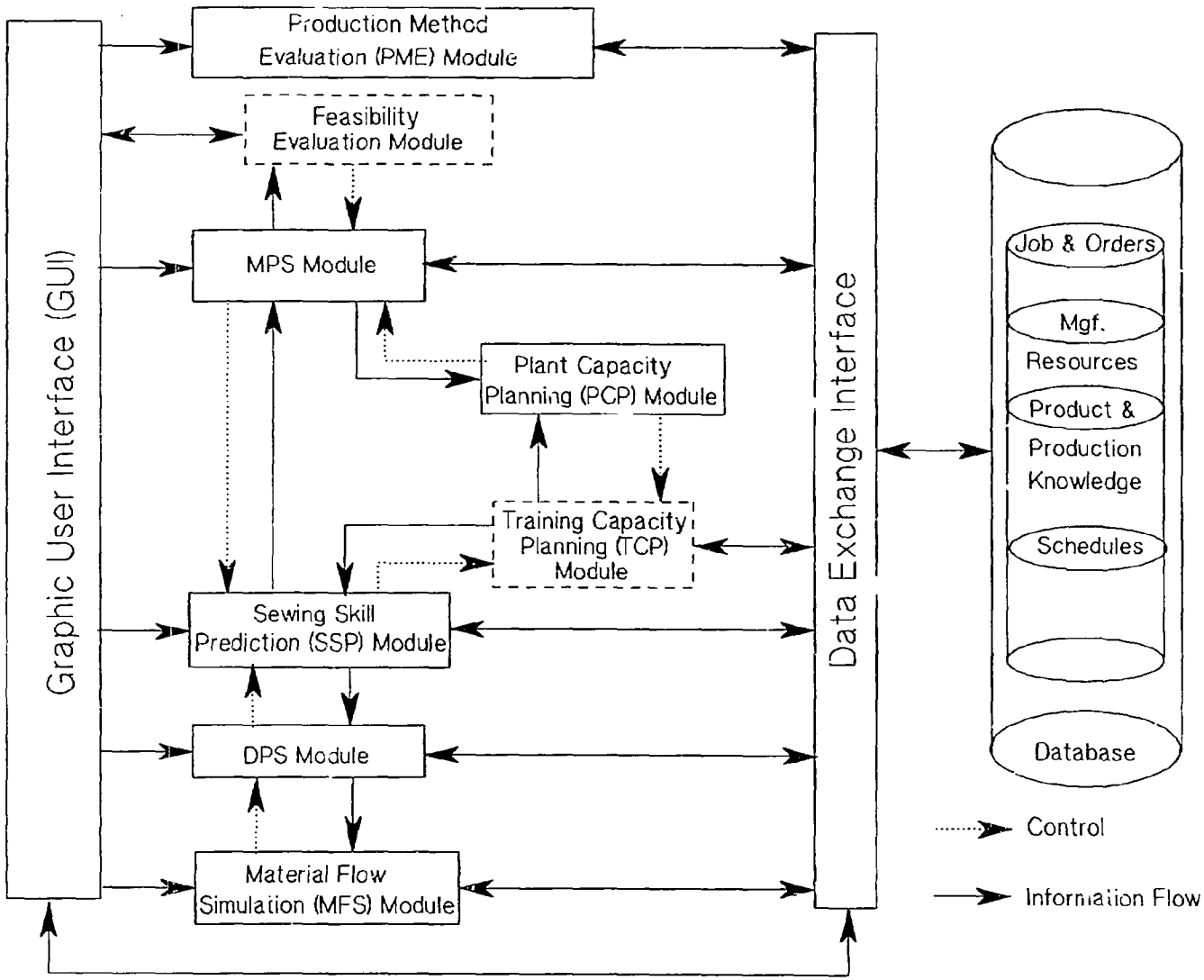


Figure 1: Proposed Apparel Production Planning & Scheduling System Architecture

regular employee turnover. External resources to be considered include subcontracts, overtime, and new hires. When the need for training arises due to new hire and/or worker re-location, the training capacity scheduler would ensure the adequacy of the training facility.

Typically a master production scheduler is run weekly or whenever a new order is being entertained. The detailed production scheduler (DPS) is applied to detail an updated weekly master production plan into a daily production schedule and specify a worker assignment plan. The material flow simulator is applied to validate the detailed production plan and worker assignment in a simulated production environment. With simulated production results, the manager then has the opportunity to modify workers' assignments before the plan is released. The database and the database manager is responsible for related data storage and integrity to support the above planning and scheduling functions.

2.2 Database Design

A database management system has many merits over the data file approach such as data independence, data integrity, data security, and economy of memory scale [38]. Since the proposed system needs to deal with a great amount of data, it could easily get into a mess if no database is used to store and maintain the data. The database design involves the development of a data base structure for the intended application. The database design process is outlined in Figure 2. The process consists of four major steps: requirement formulation and analysis, conceptual design, implementation design, and physical design. The actual design of the proposed database system is presented in Chapter 3.

There are four basic types of database management systems: relational, hierarchical, network, and indexed. The relational database is selected to implement the database system in this study. Vasta [38] summarized the pros and cons of these databases. The main disadvantage of a hierarchical database management system is that a hierarchical data structure is required, otherwise the access to data might be very time consuming. The disadvantage of a network database system is that the structure of the database is incredibly complex. Navigation through the database requires a lot of knowledge. The disadvantages of an index database management system are: 1) some system operations can be time consuming and 2) a lot of memory is required to store the index file. A relational is advantageous in: 1) easy to understand and master, 2) little need for information to be duplicated between files,and 3) relatively small memory space required for storage of data. The restructure of files is rarely needed, if a good structure is developed at the beginning. However, it may be time consuming in joining operators to work, during an application.

2.3 Graphic User Interface Design

The user interface is desired to provide a hierarchically structured menu system such that the user is guided interactively to manage job orders, manufacturing resources, product and production knowledge information, browse schedule information, and run the application modules provided by the system.

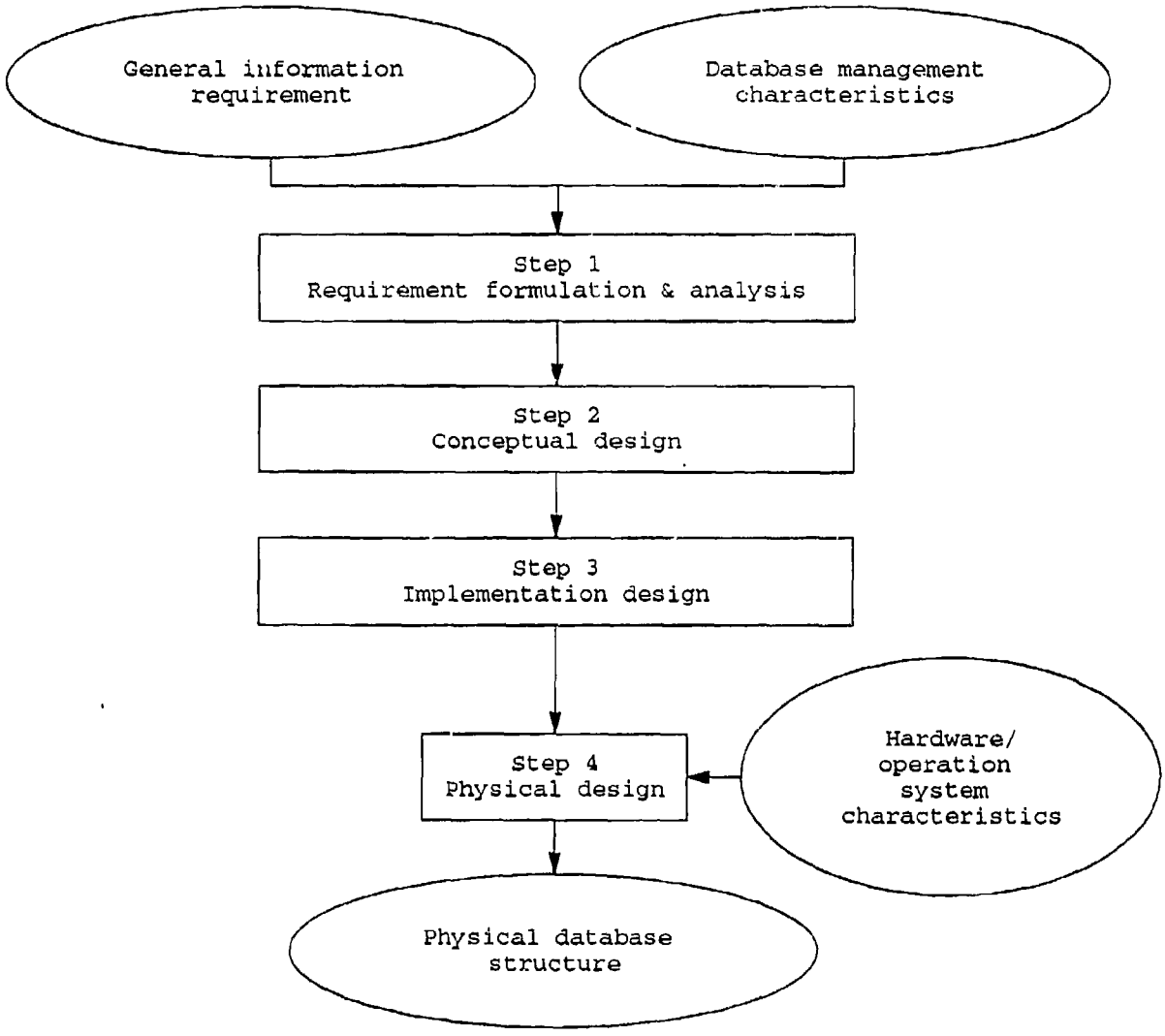


Figure 2 : Database Design Process

2.4 Sew' ; Skill Prediction

An accurate tool for sewing skill prediction is essential to apparel production planning because the capacity of an apparel plant varies significantly, depending on the job orders received, operators' sewing skills, and intended worker assignment. Many researchers have suggested means for estimating plant capacity and incorporating the effect of training requirements in production planning and scheduling [17,19,31,33]. A variety of learning curve equations are summarized in Smith [36]. These equations, however, are too simplistic and are not accurate enough for apparel production planning. A more comprehensive and apparel specific skill prediction model is proposed by Racine, Chen and Swift [32]. It consists of three separate regression models for predicting sewing skills in different situations. The first model considers the situation in which the operator does not have prior sewing experience. The second model considers the situation in which the operator is being transferred to a new operation. The third one handles the situation in which the operator is being shifted to an operation which the operator has prior experience of. The predictors used in these models include 16 operation attributes, 3 operator attributes, and some additional factors such as the similarity between the operations involved, the skill level before shifted, and the number of days on the operation.

This study continues the above theoretical effort to substantiate the first skill prediction model. The objective is to effectively predict the sewing skill of a worker at any given sewing operation type at any time. The prediction model focused in this study consists of two mathematical functions. The first is a non-linear function which captures the worker's sewing skill at an operation type according to the number of training days on the operation. This function characterizes each operation type with three parameters. The second function consists of a linear equation set which collectively predicts the worker's skill on a daily basis with 17 operation attributes. With the model, a more accurate estimate of sewing skill and plant capacity is expected and therefore a more realistic production plan can be generated. Although workers are assumed to have prior sewing experience and only operation related factors and training duration are considered, the skill prediction model should be able to make reliable short-term predictions. Since only immediately short term predictions will be used to generate actual production assignments, the potential deviation and its impact to system performance are minimal. The model construction procedure is summarized below:

1) collect production and operation data from apparel manufacturing firms.

2) digitize the time-phased worker's efficiency curves.

3) estimate the three coefficients of the non-linear function for each data curve.

4) calculate the expected daily skill levels for the first 60 days, using the non-linear function substantiated with the data curve.

5) manually examine each data point and remove those which obviously do not fit, according to the expected daily skill levels.

6) establish a matrix to store the attributes of each operation and the expected daily skill level of each operator in each data point for the first 60 days of training.

7) estimate the coefficients of the 17 operation predictors for each of the first 60 days of training.

8) validate the sewing skill prediction model, using a separate set of data.

In the following two sub-sections, the non-linear and liner functions are introduced. The substantiated prediction function is presented in section 2.4.3. The validation is provided in section 2.4.4.

2.4.1 Non-linear function

The first function of the statistical model is an algebraic power function, which is used to capture the curvature of the learning curve and intended to relate operator's sewing skill proficiency to the number of training days on a given sewing operation. It is adopted for its following desired properties:

1) the initial efficiency is zero,

2) the maximal efficiency levels off after a sufficient length of training, and

3) the learning curve varies, according to the operation type.

The non-linear model is expressed below:

$$E_{jd} = A_j \cdot [1 - \frac{1}{(1 + C_j \cdot d^{B_j})}]$$

where:

E - the expected efficiency on operation j on the d-th day of training.
d - the number of days in training on this operation.

This non-linear function contains one independent variable and three parameters (A, B, and C). The parameter A is designed to capture the maximal efficiency at the operation. The parameters B and C are used in this function to control the curvature of the learning curve. The model originated by Racine, Chen, and Swift [32] does not have the parameter C and has the parameter B outside the parenthesis instead. In addition, Rather than estimating the parameters with three independent linear functions, these three parameters for a specific operation type can be simultaneously estimated through a sequence of regression analyses.

2.4.2 Linear regression function

The second learning model is a set of linear functions, of which each separately relates the skill efficiency to the 17 operation-related predictors of a specific sewing operation type on a given day. This function includes an additional operation factor, called "hand fold," to define its operation difficulty. The generic form of this linear function is expressed as below:

$$E_{jd} = a_0 + \sum_{i=1}^{17} a_i \cdot x_i$$

The SX linear regression package is used to run the linear regression model. Since only operation-related factors are considered in the model, potential variations in operators can not be differentiated.

2.4.3 Substantiation of the prediction function

A set of 589 data points were collected from 12 apparel firms for the development and validation of the model. The most difficult task in the data collection process was to search through mountainous data piles to identify sewing workers who had no prior sewing experience. Only the production records of these workers are applicable; however, they were not plentiful. Two types of data were sought after. The first type is daily production records of operators/trainees assigned to each operation type. It includes the number of days at the operation type and the operator's daily efficiency at the operation type. The second type of data is the attributes of each operation type. This data are extracted by observing trained operators performing the operation type and/or by engaging a discussion with plant personnel.

The non-linear algebraic power function was defined as a function of the number of days on training at the given operation. The least square method was used to estimate the three (A, B, and C) coefficients from the 589 data points at 95% confidence level. After the coefficients for each data point were estimated, the model was used to project expected skill on each day at the operation type. The expected daily skills form a smoother skill progression curve and are used to visually evaluate the fitness of each data point. Forty six data points were determined as unfit and discarded in this process. The expected daily skills and the 17 operation-related attributes of the remaining data (543 data points in total) were then used as the basis for developing the linear model.

The linear model takes into account the 17 operation type related variables and is constructed by using the expected daily skill level originated from each data point. The predictors used in the linear function are: piece rate, number of stops, number of slowdowns, number of match points, speed control, plaid match, back tacking, stops with prediction, ease requirement, folder requirement, position control of stitches, ply alignment, fabric sewability, align and position difficulty, dispose difficulty, difficulty in attaining quality standard, and hand fold. The least square method was used to estimate the coefficients of the predictors for each operation type and each day.

The mathematical representation of the linear function for estimating predictors' coefficients on dth day is presented below:

$$\begin{bmatrix} 1 & X_{1,1} & X_{1,2} & \cdots & X_{1,17} \\ 1 & X_{2,1} & X_{2,2} & \cdots & X_{2,17} \\ . & . & . & \cdots & . \\ 1 & X_{j,1} & X_{j,2} & \cdots & X_{j,17} \end{bmatrix} \cdot \begin{pmatrix} a_{0,d} \\ a_{1,d} \\ . \\ a_{17,d} \end{pmatrix} = \begin{pmatrix} E_{1,d} \\ E_{2,d} \\ . \\ E_{j,d} \end{pmatrix}$$

The coefficients $a_{0,d}$, ..., $a_{17,d}$ may be obtained by:

$$\begin{pmatrix} a_{0,d} \\ a_{1,d} \\ . \\ a_{17,d} \end{pmatrix} = \begin{bmatrix} 1 & X_{1,1} & X_{1,2} & \cdots & X_{1,17} \\ 1 & X_{2,1} & X_{2,2} & \cdots & X_{2,17} \\ . & . & . & \cdots & . \\ 1 & X_{j,1} & X_{j,2} & \cdots & X_{j,17} \end{bmatrix}^{-1} \cdot \begin{pmatrix} E_{1,d} \\ E_{2,d} \\ . \\ E_{j,d} \end{pmatrix}$$

where E_{jd} - the expected skill level on operation type j on the dth day training.

The SX statistics software package was used to run the linear regression model. The coefficients of the 17 predictors for the skill prediction on the first 60 training days at any given operation were computed. The coefficients for the first five days are listed in Table 1. For example, the linear function for the first day of training can be written as:

$$\begin{aligned} E_{j1} = {} & 17.519 - 0.0495X_1 - 0.632X_2 + 0.125X_3 + 0.314X_4 - 0.495X_5 \\ & - 0.267X_6 - 0.476X_7 - 0.449X_8 - 0.343X_9 - 0.580X_{10} + 0.196X_{11} \\ & - 0.145X_{12} - 0.323X_{13} - 0.0186X_{14} - 0.209X_{15} + 0.106X_{16} - 0.381X_{17} \end{aligned}$$

The prediction of a given operation beyond the first 60 days can be extended by substantiating the three parameters of the non-linear model using the predicted skills in the first 60 days.

2.4.4 Model Validation

These models were tested with six data points, using the goodness-of-fit test [39]. The

Table 1 : Coefficients for the Skill Prediction

Pred. Coef.	D1	D2	D3	D4	D5
a_0	17.519	31.975	43.444	52.471	59.71
a_1	-0.0495	-0.1577	-0.3354	-0.466	-0.5187
a_2	-0.6315	-0.3971	0.0158	0.3458	0.5183
a_3	0.1253	-0.0705	-0.3492	-0.6319	-0.8936
a_4	0.3139	-0.160	-0.72	-1.2348	-1.6489
a_5	-0.495	-0.7342	-0.8115	-0.8261	-0.8113
a_6	-0.2667	-0.3235	-0.3264	-0.3109	-0.2938
a_7	-0.4764	-0.8128	-1.0506	-1.1859	-1.254
a_8	-0.4495	-0.3752	-0.09315	0.1982	0.4282
a_9	-0.3429	-0.6057	-0.8228	-0.9725	-1.0618
a_{10}	-0.5798	-1.069	-1.4874	-1.8168	-2.0635
a_{11}	0.1959	0.3025	0.4671	0.6223	0.7471
a_{12}	-0.1454	0.01374	0.1268	0.2145	0.2754
a_{13}	-0.3229	-0.8443	-1.304	-1.6243	-1.8385
a_{14}	-0.0186	-0.1484	-0.2189	-0.2793	-0.3396
a_{15}	-0.2094	-0.4539	-0.6855	-0.8733	-1.0288
a_{16}	0.1058	0.1455	0.1887	0.2289	0.2708
a_{17}	-0.3806	-0.8578	-1.3369	-1.74	-2.0459

six data points are different from the 589 data sets which were applied to build the model. A comparison of one data curve with the theoretical skill predictions is shown in Figure 3. The linear prediction model was accepted at 99.5% confidence level. In addition, the absolute variation average between the six data curves and the predicted values is 8.16%, which seems reasonable from a practical point of view. Note that, however, at 99% confidence level, the coefficients of determination (R^2) of these daily prediction models are generally between 0.2 and 0.3, which are considered to be rather low from an engineering point of view. The reasons might have been the skewed distribution of the data collected and the variations in individual operators.

As an application example, an operation type (called Set Seam), which is a data point collected from the Lee's Manufacturing Company, has its 17 operation attributes recorded as 2.551, 1, 1, 2, 10, 0, 0, 0, 0, 10, 5, 10, 5, 5, 5, 3, and 0, respectively. Plugging these values in the linear function using the coefficients for the first five days listed in Table 1 yields an expected skill proficiency at 3.85%, 7.60%, 10.95%, 14.16%, and 17.23%, respectively. By applying the daily skill predictions for the first 60 days, the three (A, B, and C) coefficients of the non-linear function can also be estimated. For this particular application, the coefficients (A, B, and C) are estimated to be at 117.54, 1.1224, 0.02454, respectively. With these coefficients, the non-linear model can predict the expected skill level for this operation on any given day, within and beyond the first sixty days. For example, the skill levels for a new hire at this operation type on the sixtieth and the 100th day are expected to be 83.27% and 95.4%, respectively.

In summary, the model can make reasonable prediction (within 10% of variation), according to the goodness-of-fit test. It can be applied to any type of sewing operations. The coefficient of determination (R^2) of this model, however, is lower than anticipated. It could be improved by using a more balanced operation and production data and by including operator related factors. The validity of this model for predicting operators's skills with prior sewing experience has not been established. To predict the start-up skill on an operation of prior experience, the model may have to take into account additional factors such as the similarity between these operations and the skill level on the current operation type.

2.5 Master Production Scheduling

Master production scheduling generates a weekly production for each job order within a planning horizon, which can vary from a few weeks to several months, depending on job orders, manufacturing lead time, demand forecast, material delivery, and the desired ability to respond. Master production scheduling is a critical interface between marketing and manufacturing functions, because it links customer service directly to production resources [12] and consolidates business and production activities into a single and coordinated plan for all operations. Master production scheduling is a continuous process as many system parameters constantly vary and affect business, planning and production activities in the company. Any apparel master production scheduler must be able to quickly re-schedule its production plan to accommodate market dynamics and fosters the growth of worker's sewing skills and plant capacity.
Few MPS techniques, however, are available for make-to-order production, while voluminous

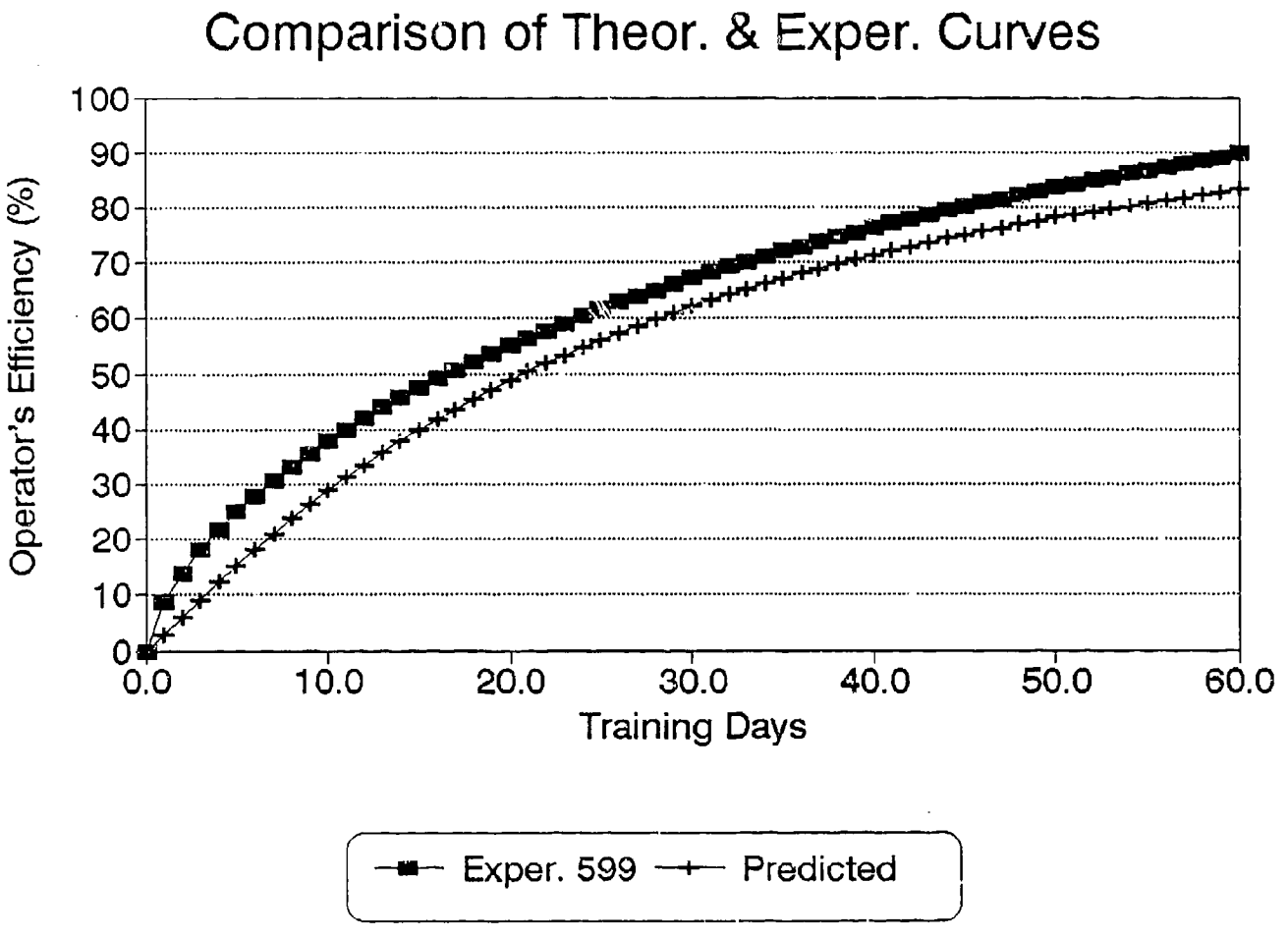


Figure 3 : Skill Progression Curves

research papers are available in the literature, promoting new MPS techniques for make-to-stock production. Some make-to-stock MPS procedures consider resources constraints and schedule production plans for items below end-item products by applying MRP and BOM data. Some of these MPS techniques may be applicable to both production methods [18]. However, the problems in the two environments are so different that these techniques are in general not mutually applicable. In fact, master production scheduling in a make-to-order production environment is much more difficult [25]. Most apparel firms, especially small and medium sized, apply strictly make-to-order practices and have to rely on experience to manage plant resources, not for momentary optimum and balance but long-term capacity gains. The concept is presented in Chen, Racine, and Swift [6]. A mathematical treatment of the MPS problem is presented in [7]. Mathematical models, however, cannot easily capture the assembly operation relationships and more critically with linear variables. The need for keeping workers at the same operation type to foster skill growth cannot be explicitly captured either. In addition, such a mathematical model can not be solved efficiently for a practical size of scheduling problem for the apparel industry, where a large volume of garment entities constantly move through the shop floor quickly.

As described above, most existing MPS techniques are focused on make-to-stock or "pseudo" make-to-order production. Even though some MPS approaches took into account the capacity constraints, most of them still remain at rough-cutting level and ignore time-phased skill learning effect on plant capacity. The proposed MPS procedure attempts to ensure adequacy of resources for the scheduled production by analyzing job orders and worker's skills at operation type level and taking into account the time-phased sewing skills and plant capacity, which can be managed to grow. To prepare a master production schedule, the following information are needed: existing and new orders, available resources, and product and production knowledge. We further assume that:

- The quantity of each job order, its product structure, operation type and standard processing time of each job order are specified.
- A skill prediction model is available for each operation type and can be effectively characterized by a set of operation attributes.
- The current skill level of each worker at each operation type is known.
- Available alternative resources are specified for the entire planning horizon.

The proposed master production scheduling procedure is outlined in Figure 4. It generates a master production schedule in three stages in a week-by-week fashion. In this procedure, existing orders are updated and validated first because commitments have been made and workers may have begun to work on the orders. The workers current working on existing orders have a higher priority to remain on the job. At the second stage, new job orders are then posted to the master production plan with the remaining resources. The procedure stops if all new arrivals can be scheduled within their due date with existing in-house resources. Excessive resources may

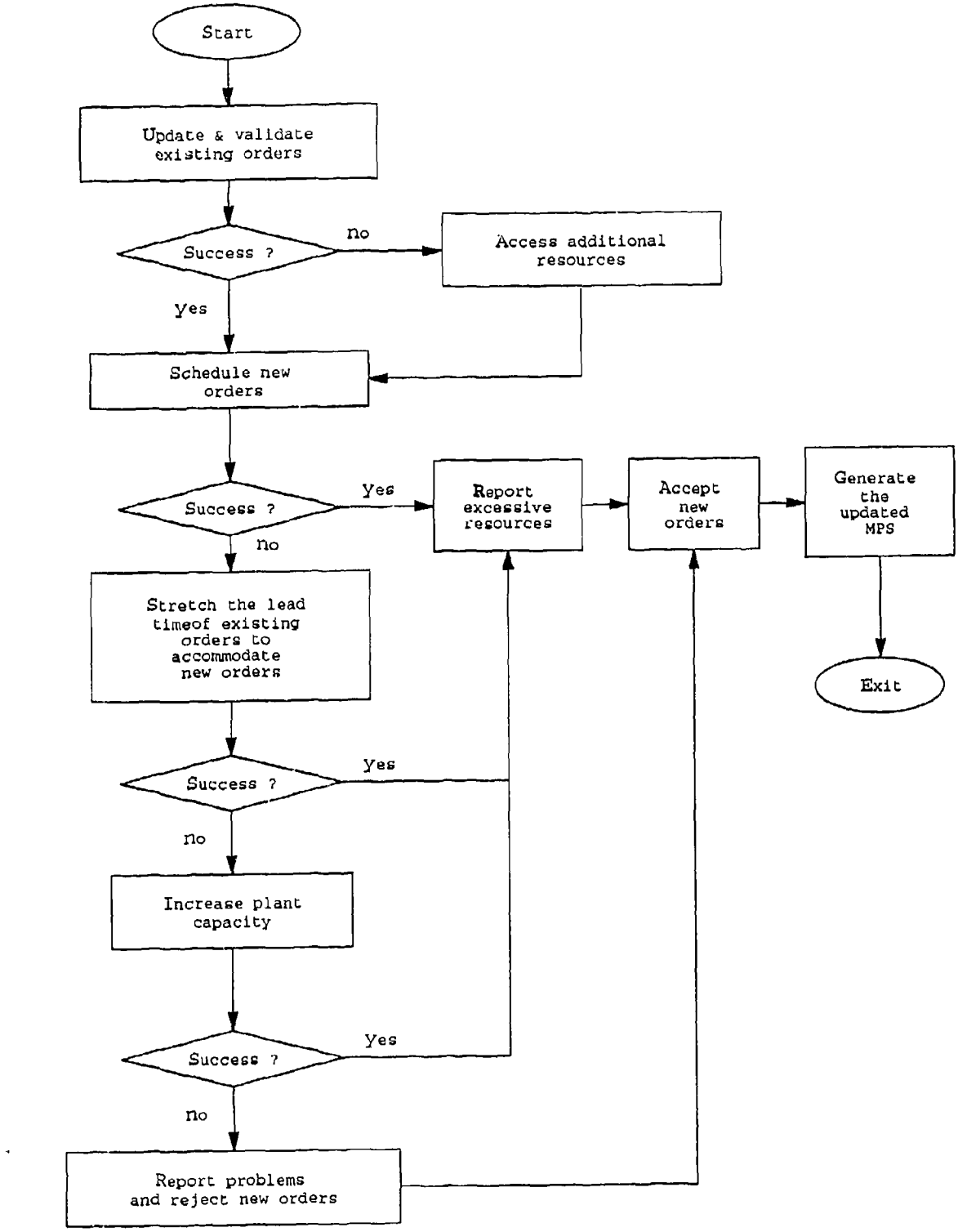


Figure 4 : Master Production Scheduling Process

be reported at this time, if they exist. If some jobs can not be scheduled to meet their due date, the procedure continues onto the third stage, at which workers may be pulled out from an existing job order and, if without success, external resources are then considered.

At the first stage, the procedure starts by updating and validating existing orders. First of all, the production quantity actually accomplished in the last week for each existing order is fed to the system to update job status. A validation procedure is then activated to ensure the current manning level for each job order can finish the remaining workload before or on its due date. When a potential problem is detected, the amount of additional resources required is estimated. This problem occurs only when there is a significant employee turnover, because the underlying assignment principles for the proposed approach are that: 1) it initializes the production level for each order by averaging out the production over the planning period for each job order and then 2) it attempts to maintain the same level of workforce over the production period. Due to the expected learning effect, there should be a gradual and healthy growth in the throughput and thus result in a shorter lead time.

At the second stage, new orders are scheduled with remaining resources, which become available when workers are released from a job just being finished or new employees are hired. The approach to scheduling of new jobs is depicted in Figure 5. All new jobs are first ranked according to a job urgency index, which is based on the due date of each job. A production level is then initialized for each job order by averaging out the order quantity over the available time. The production level is initially set according the formula below:

$$Q_{jw} = \frac{Q_j}{(W_d - W_c)}$$

where,

Q_{jw} - initial weekly production quantity of the job j in week w,
Q_j - the total production quantity of job j, and
W_d, W_c - the calendar week of due date and the current calendar week of job j, respectively.

At the third step, starting from the first planning week, the estimated production in the week is decomposed into work contents by operation type for all new jobs:

$$R_{wp} = \sum_{j=1}^{J} Q_{jw} \cdot S_{jp} \quad for\ p = 1 \ldots P$$

where

R_{wp} - the required capacity of the operation type p in the week w,
S_{jp} - the standard processing time for operation type p of job j.

An urgency index is then used to ranking each operation type according to the estimated workload for each operation type and total remaining workforce available for each operation type.

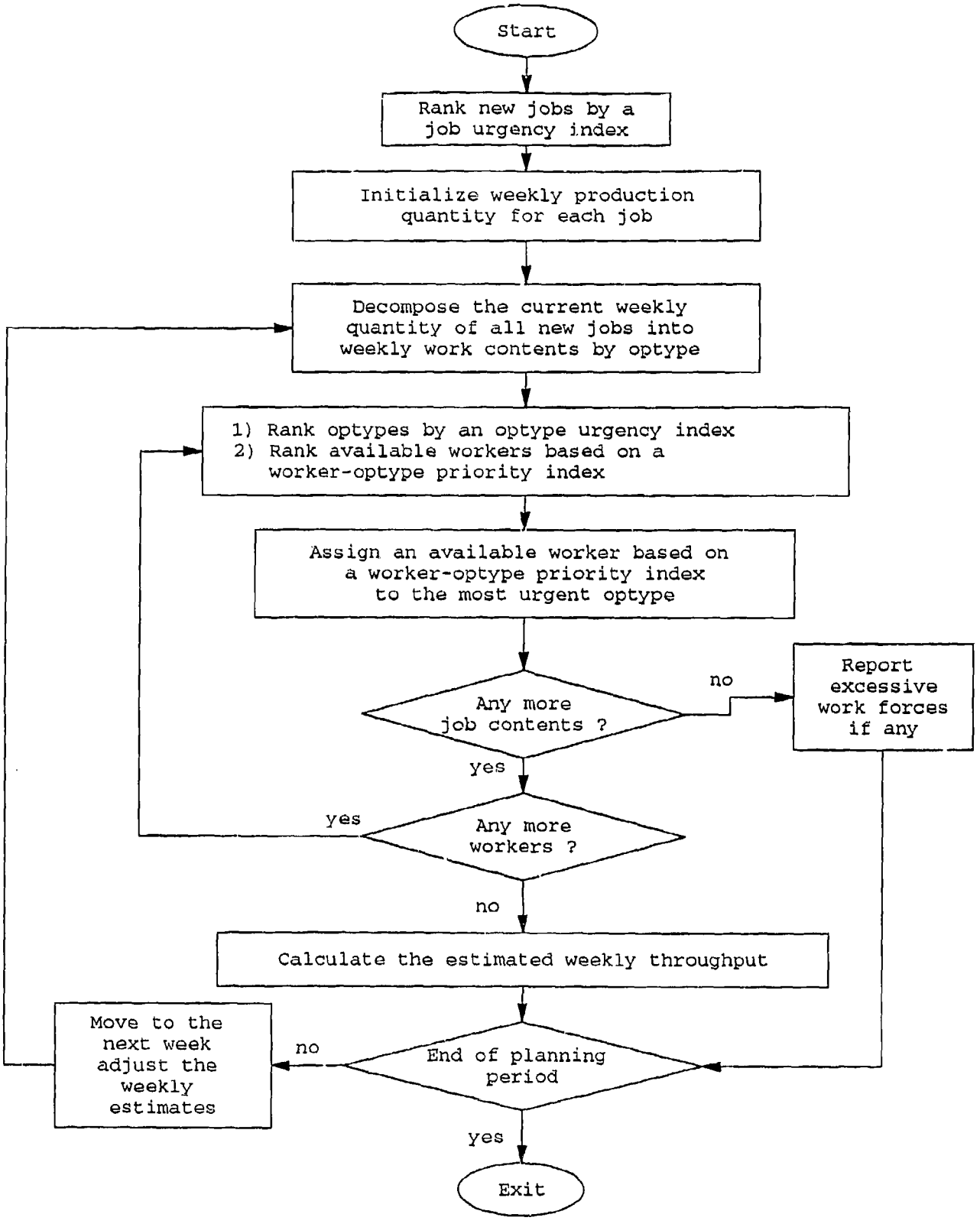


Figure 5 : New Job Loading and Worker Assignment Procedure

The index is established as:

$$U_p = \frac{\sum_{j=1}^{J} Q_{jw} \cdot Sjp}{\sum_{k=1}^{K_p} \sum_{d=1}^{D_k} (h_{kd} \cdot e_{kdp})}$$

where

U_p - the urgency index of operation type p,
h_{kd} - the working hours of worker k on dth working day in the current week w,
e_{kdp} - the skill level of worker k on operation type p on the dth day,
K_p - number of available workers to be assigned to the operation type p,
D_k - the total working days of worker k in week w.

In the meantime, available workers are also ranked according to a worker-to-operation type assignment priority index, which combines worker's skill level at each operation type and prior assignments, among other factors. The most urgent operation type is assigned with the highest ranked operator, based on the two indices. The updated urgency index for operation type is shown as below:

$$U_p' = \frac{R_p - \sum_{d=1}^{D_k} (h_{kd} \cdot e_{kdp})}{\sum_{k=1}^{K_p} \sum_{d=1}^{D_k} (h_{kd} \cdot e_{kdp}) - \sum_{d=1}^{D_k} (h_{kd} \cdot e_{kdp})}$$

where

U_p' - the updated urgency index for operation type p.

The procedure repeats until all the estimated production level for the week is reached or all available workers are assigned. The substantiated weekly production is then estimated as below:

$$Q_{jw} = \min\left(\frac{\sum_{k=1}^{K_{j1}} h_{kd} \cdot e_{kd1}}{T_{j1}}, \frac{\sum_{k=2}^{K_{j2}} h_{kd} \cdot e_{kd2}}{T_{j2}}, \ldots, \frac{\sum_{k=p}^{K_{jp}} h_{kd} \cdot e_{kdp}}{T_{jp}} \right)$$

where

K_{jp} - number of workers assigned to operation type p on dth day in the week w,
T_{jp} - percentage of standard processing time spent on operation type p to the sum of standard processing time of job j.

The estimated value is used to update the remaining production quantity for the following

weeks. The adjusted production level for the subsequent weeks is established with the following equation:

$$Q_{jw}' = \frac{Q_j - Q_{jw}}{w_d - w_c - 1}$$

where

Q_{jw}' - the adjusted production level.

The procedure then continues into the next week until the last week of the planning horizon. At the end, a check is made to ensure all new jobs are scheduled. If due dates are not met, the MPS procedure is carried on to the third planning stage.

At the third stage, the procedure first searches for possibly available resources from existing orders. If an existing order is scheduled to finish before its due date, it is called "stretchable", meaning that by stretching its makespan to its due date, some workforce can be released for other use. If the released workforce cannot meet the due date of new jobs, the MPS procedure continues to search for additional resources. At this point, an increase in plant capacity is considered with subcontract, overtime, and new hire as its options. The priority and capacity limit of each are pre-set by the system. If available external resources cannot meet the need, the problem is reported and new jobs are rejected according to a given priority. The output of the MPS procedure includes excessive workforce report, status of job orders, and an updated master production schedule for the entire planning horizon.

2.6 Detailed Production Scheduling

The proposed detailed production scheduling procedure takes the production plan for the immediate week from the master production schedule and generates a daily production and worker assignment plan at workstation level. It prepares the plan only for the coming week, because changes always occur which quickly outdate any detailed production plan of longer planning period. Detailed production scheduling includes two tasks: loading and sequencing [26]. Since no assignment is done at machine and workstation level during master production scheduling, loading assigns jobs and workers to workstations. Sequencing orders jobs at each workstation so as to achieve some goals such as maximizing job throughput or minimizing the job lateness, among other. In apparel manufacturing, sequencing is usually left for the shop floor. Goodwin [15] made the following assumptions to simplify the loading problem in an assembly shop environment:

1) Each workstation performs one operation.
2) The processing time is a linear function of order size.
3) The labor supply is unlimited.
4) All workers are perfectly efficient.

With these assumptions, the loading task becomes balancing the production flow across its manufacturing process. A balance factor was defined as a processing time function to establish the work-load balance between the sequential workstation. Russell and Taylor [35] consider the dual constraints of machines and worker resources, but loading of jobs to machines is determined, assuming that routing of each job was predetermined so that every operation of jobs has an equal probability to be performed at any machine [35]. The longest queue of items waiting to be processed was used to allocate workers to machines. Meredith [26] introduced an index method to deal with job loading, which calculates the "efficiency index" for each job on workstation, and load the workstation with those jobs that have the best indices for that workstation. The assignment of workers to workstations is not mentioned. Many sequencing rules such as SPT, FCFS, BDD and slack time per operation have been proposed and studied mainly for job shops in numerous studies including [1,2,10,11,12,15,20,29,35]. While there is no single best rule for all job shop situations, these rules in general cannot be applied to the apparel production problem. Typically they ignore learning effect and calculate static machine capacity. The study of sequencing rules for apparel assembly shops is not available.

A mathematical treatment of the detailed production planning problem has been presented in a previous report. Two theoretical DPS models for this problem are available in Chen [7]. The remaining of this section will be focused on the proposed DPS procedure. It is designed to take operation types and worker assignment from the MPS and breaks them down to workstation and job/operation level. The DPS procedure is carried out in two planning stages. An initial production schedule is established at the first stage and then improved at the second stage. A heuristic procedure taking into account all relevant factors is developed for generating an initial detailed schedule. A schedule improvement algorithm based on the simulated annealing concepts is developed to improve the initial schedule.

2.6.1 Generation of the initial production schedule

The proposed procedure for generation of an initial detailed production schedule can be divided into two stages. At the first stage, jobs are loaded to workstations. Each worker is assigned to a workstation and a job. The loading procedure at the first stage is depicted in Figure 6. The procedure begins with ranking new jobs based on a job urgency index. The currently most urgent job is selected first for loading. To decide which workstation to be loaded, the current remaining loading level at each station is calculated. The currently targeted operation (according to the operation precedence) of the most urgent job is assigned to the feasible but least loaded workstation. The loading activity for the current job continues until all its operations are assigned. The job loading process continues until all new jobs are assigned. Since the input from the master production schedule is a weekly production quantity, the loading process is also done according to weekly capacity of each workstation. The process assumes that the weekly quantity set by the MPS is feasible at operation type level. It is assumed that a job can visit any workstation only once. At this stage, the remaining workstation capacity is calculated based on standard workers.

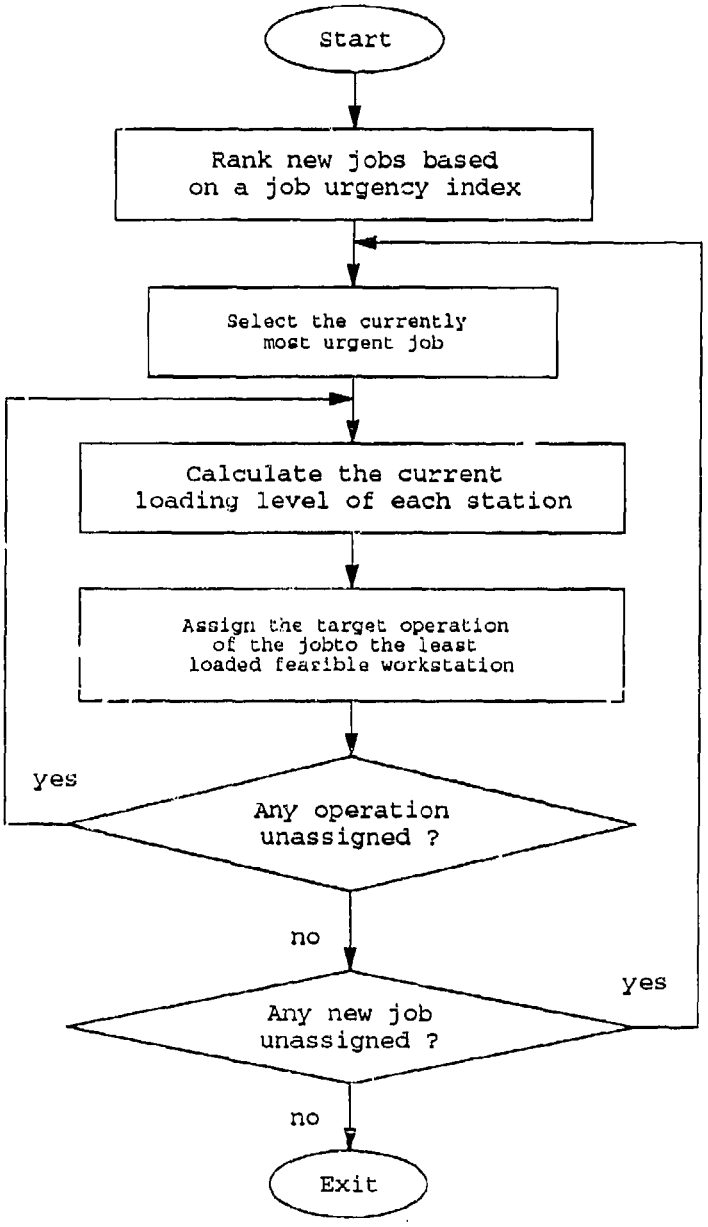


Figure 6 : Loading Workstations with New Jobs

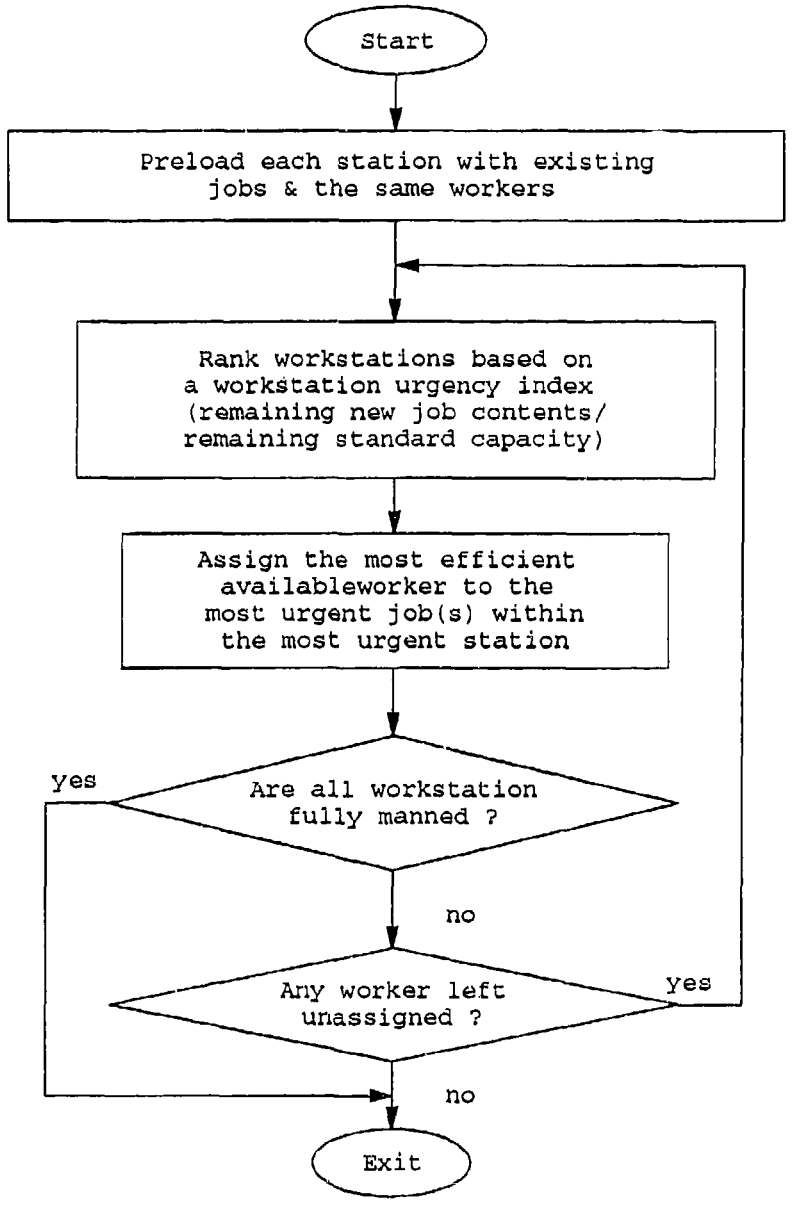


Figure 7 : Worker Assignment to Job(s) and Workstations

The assignment procedure at the second stage is depicted in Figure 7. The procedure starts with re-loading each operation of all existing jobs to the same station and with workers. A workstation urgency index is calculated based on the ratio of the remaining new job contents to the remaining standard capacity. The most efficient and available worker is then assigned to the most urgent job within the most urgent workstation. The assignment process continues until all workstations are fully manned or no more workers are available. After each assignment in the process, the workstation urgency is updated with the projected weekly efficiency of each worker assigned to the workstation.

2.6.2 Schedule Improvement

A schedule improvement procedure is proposed to improve the initial detailed production plan, using the simulated annealing (SA) concepts. The procedure generates alternative plans from the initial plan by shifting workers between operations and workstations to improve the throughput while meeting due date and other constraints. Simulated annealing has demonstrated a prominent ability in solving a wide variety of combinatorial optimization problems [5,9,21, 22,24,40]. The SA concepts have been applied to the job shop scheduling problem by Van Laarhoven, Aarts, and Lenstra [24]. Rolling [34] further studied a range of annealing schedule parameters to identify an optimal parameter set for the job shop problem.

Simulated annealing is mathemadically described as the generation of a sequence of homogeneous Markov chains. The simulated annealing algorithm can be viewed as a sequence sof Metropolis algorithms evaluated at a sequence of decreasing values of the control parameter. It can be described as follows:

1) Initially the control parameter is set at a high value.

2) A generation mechanism is defined so that, given a configuration i, another configuration j can be obtained by randomly selecting an element from the neighborhood of i (the later corresponds to the small perturbation of the Metropolis algorithm).

3) Let ΔC_{ij}= C(j)-C(i), then the probability of configuration j to be the next configuration is given by 1, if $\Delta C_{ij} \leq 0$, and by $\exp(-\Delta C_{ij}/T)$, if $\Delta C_{ij} > 0$ (the Metropolis criterion). The acceptance criterion is implemented by drawing random numbers from a uniform distribution on (0,1) and comparing these with $\exp(-\Delta C_{ij}/T)$.

4) This process continues until a threshold value for the control parameter is reached.

An unique feature of simulated annealing techniques is that it allows neighborhood moves which increase the value of the cost (or objective) function to avoid being trapped in a local optimum. The probability of accepting a move which causes an increase of δ in the objective function is called the "acceptance function" and is normally set to $\exp(-\delta/T)$, where T is a csontrol parameter which corresponds to temperature in the analogy with the physical annealing. A flow chart for simulated annealing is shown in Figure 8 and described below:

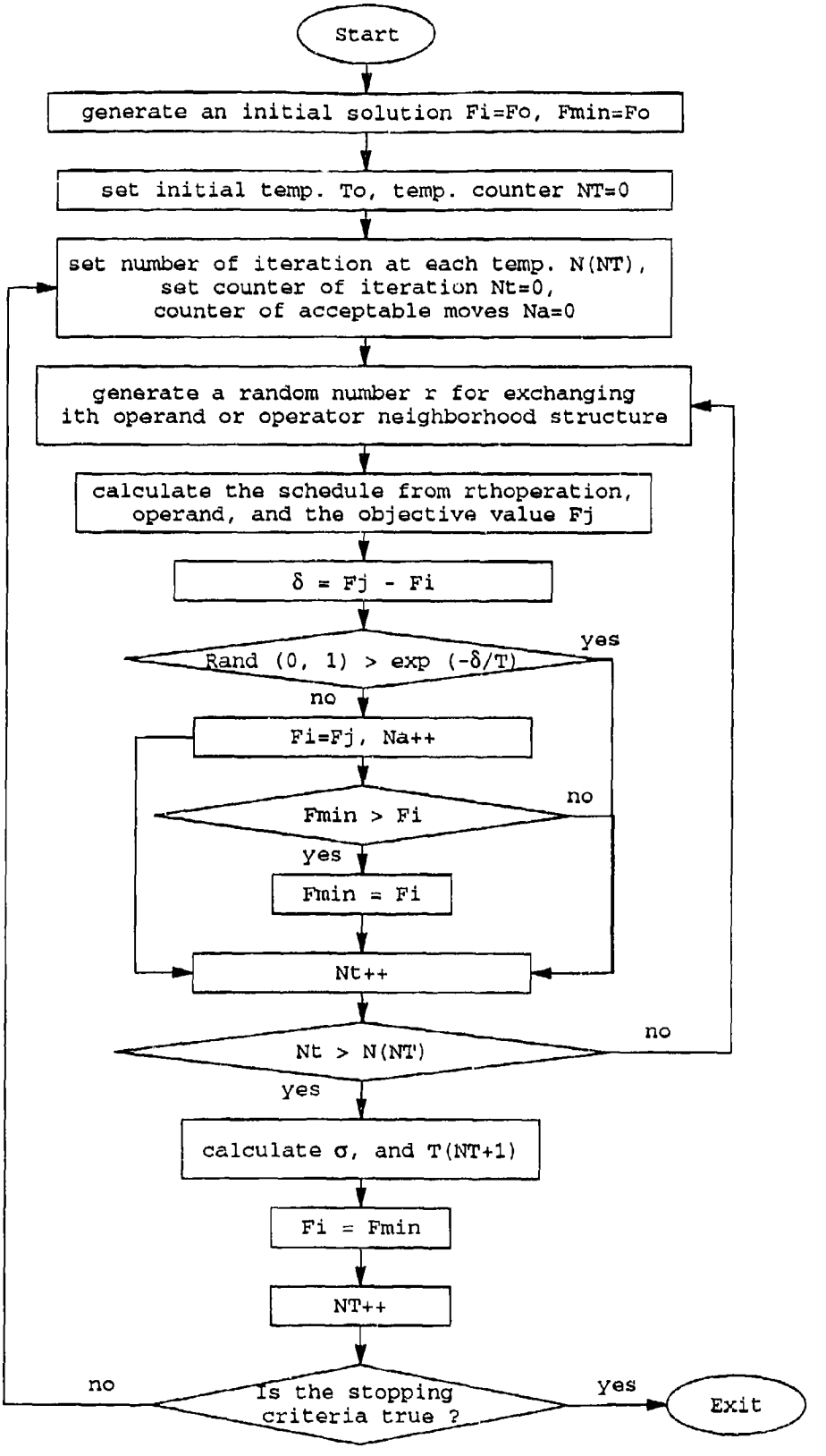


Figure 8 : Schedule Improvement Procedure

1) generate an initial solution (i.e., a detailed production plan schedule); estimate the objective function value for the initial solution, F_0; set $F_i=F_0$, and $F_{min}=F_0$; and set the initial temperature, $T=T_0$; and initialize the temperature changing counter, N_T, to 0.

2) set an iteration counter (N_t) at the temperature T to 0; set an acceptable move counter (N_a) to 0; and calculate the maximum number of iterations at temperature T to $N(N_T)$,

3) generate a random number (r) for exchanging the *i*th operand or its operator of the neighborhood structure,

4) calculate the new objective function (F_j), and let $\delta = F_j-F_i$.

5) check the condition: $rand(0,1) < \exp(-\delta/T)$. If it is satisfied, go to step 6), otherwise go to step 8,

6) let $F_i=F_j$, and $N_a=N_a+1$,

7) check $F_{min}>F_i$, if it is satisfied, let F_i=Fmin,

8) set $N_t=N_t+1$,

9) check if $N_t>N(N_T)$. If it is not true, go to step 3, otherwise go to the next step.

10) set $F_{min}=F_i$, $N_T=N_T+1$, calculate the standard deviation of the objective function, σ, and the temperature for the next iteration $T(N_T)$,

11) check the stopping criteria. If a criterion is met, stop; otherwise go to step 2.

With the proposed application, the operator and operands for the DPS solution expression are defined in the subsection 2.6.2.1, followed by a description of the SA solution expression for the detailed production plan in subsection 2.6.2.2. The generation mechanism for alternative solutions is presented in the subsection 2.6.2.3. The annealing parameters set used in the proposed schedule improvement procedure is adopted from Rolling [34].

2.6.2.1 definition of SA operators and operands for the schedule

Workstations are defined as SA operands for the proposed DPS solution improvement algorithm. The SA operators are defined to be worker oriented as below:

+ assign a worker to a workstation based on the workstation urgency; i.e., assign the most efficient worker to the most urgent workstation.

- assign an available worker to a workstation randomly.

2.6.2.2 SA expression for the schedule

With the above SA operators and operands definition, a SA solution expression of a detailed production schedule can be exemplified as:

$$\alpha = (+W1+W2+\ldots+Wi)\,P1\,(+W3+W2+\ldots+Wj)\,P2\ldots(W4+W1+\ldots+Wm)\,Pn$$

where Wi - workstation index,
Wn - operation type index.

The above expression captures each operation type in a set, each representing a complete worker-workstation assignment by the operation type. Each operand has a workstation ID and the sequence of each operand showing the urgency ranking of its workstation. The number of operands in each set indicates the number of workers to be assigned to the operation type according to the MPS module. A change to the operand sequence implies changing their urgency ranking. In the expression shown above, all operators' prefixes are initially the same as one another because workers are assigned to a workstation according to the same assignment rule. In the above expression, all workers are assigned according to their skill at the assigned operation.

2.6.2.3 mechanism for alternative schedule generation

There are two swap methods (S1 and S2) prepared for alternative solution generation. The swap method (S1) switches the urgency of two workstations. The swap is expected to have an avalanching effect on worker assignment to workstation. The swap method (S2) alters the method of selecting a worker for a workstation and as a result affects all the assignments following the selection of a workstation. The stopping criteria applied to the proposed improvement procedure include the total number of iteration, a pre-determined temperature, and a preset number of iterations in which the objective value has not been improved.

2.7 Plant Capacity Planning

Capacity planning is used in the proposed system to adjust plant capacity when available resources do not match with delivery commitments. In general, there are two types of capacity planning activities. The long-term capacity planning concentrates on the facility expansion and contraction of facility from a strategic point of view. The short-term capacity planning considers capacity alternatives from a tactic point of view. The short-term capacity planning is also applied to handle unexpected but imminent demands. The planning period for short-term capacity planning is typically less than six months, which is the focus of the proposed system.

The short-term capacity planning activity is closely coupled with production planning and scheduling activities. While capacity planning is primarily oriented toward the acquisition of

productive resources, scheduling concerns about the timing of their use. The additional productive resources are typically acquired from the following sources:

1) Increase in resources by using overtime, adding new shifts, hiring part-time workers, using floating workers, leasing workers and facilities, and subcontracting.

2) Improve resource utilization by overlapping or staggering shifts, reducing absenteeism, adding new resources, creating backlogs or demand queues.

3) Modify the demand by changing the price, modifying promotion, and negotiating new due dates.

In the proposed capacity planning approach, only three alternatives are considered: overtime, subcontract, and new hire. When the available production resources can not meet the demand, The first option is overtime. Subcontracting is considered the next option, and adding new workers is the last resort. When the overtime function is activated, the dates for overtime are specified and overtime workers are assigned to identified operation types according to the skill of available workers at the needed operation type. In the planning process, the urgency status of each operation type is established and constantly updated after assigning each overtime worker. When the subcontract option is chosen, a subcontract evaluation procedure is activated to identify subcontractor's capability in terms of the product(s) each subcontractor can handle, quantity, resource available periods, and cost structure. Once a decision is made, each affected job is modified accordingly. Subcontract quantity is recorded and so are the starting and finishing dates. When hiring option is selected, a new-hire function is activated to estimate the number and skill types required of new hire workers, and amount of training needed such that training facility and production equipment are available. The proposed plant capacity planning procedure is outlined in Figure 9 and summarized below as well.

1) Input the insufficient operation type IDs and standard times from the MPS module.

2) Activate the overtime function to identify workers available for overtime from the worker table and generate an initial worker availability list, ranked by the worker's sewing efficiency for each insufficient operation type.

3) Initialize an urgent list for all operation types with insufficient manpower and generate an urgency list according to the ratio of the required total standard time to the manpower available for overtime.

4) Assign the most efficient overtime worker to the most urgent operation type.

5) Adjust the processing time deficit and update the operation urgency list, and the worker list.

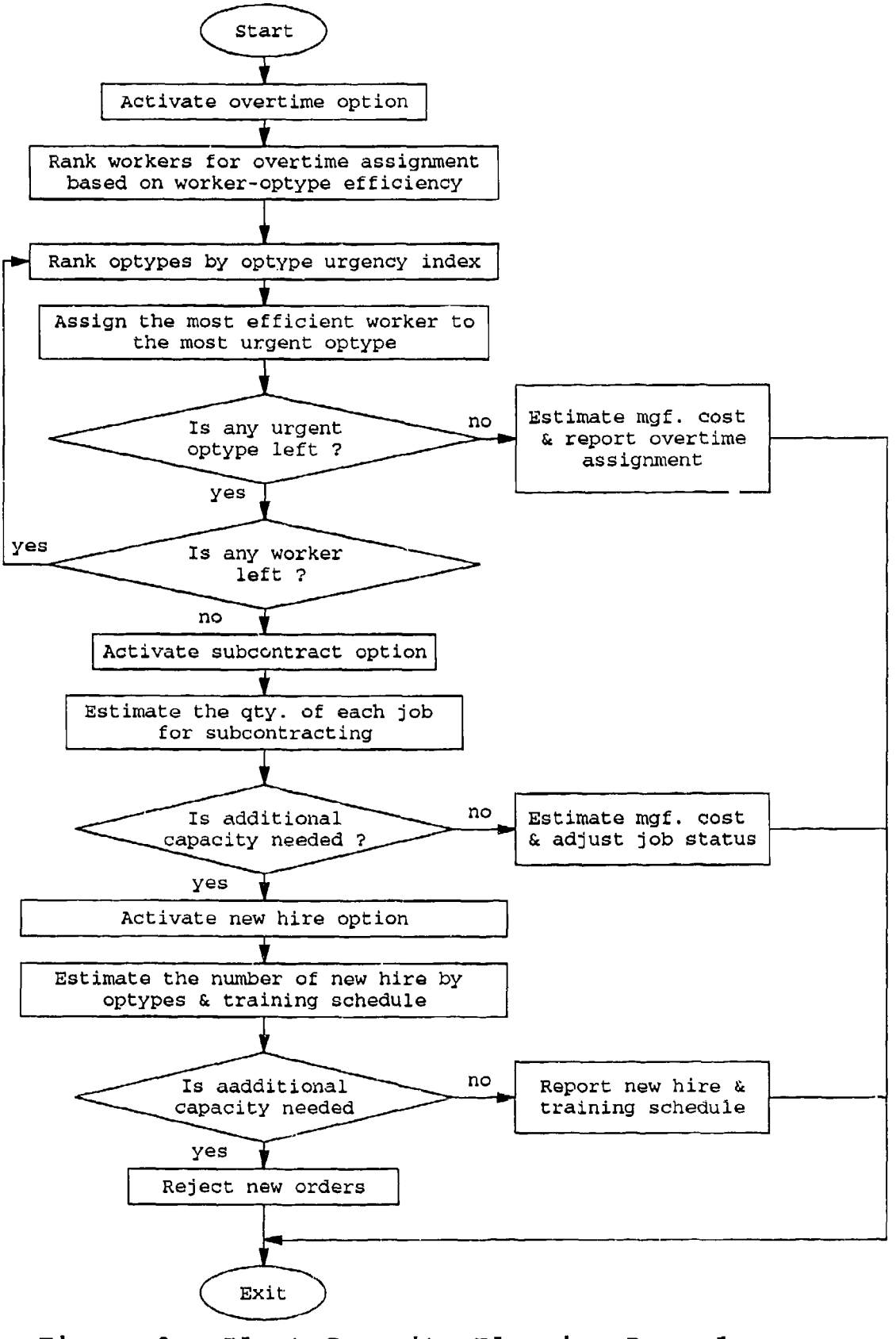


Figure 9 : Plant Capacity Planning Procedure

6) Repeat steps 4 and 5 until the urgency is resolved or no more overtime worker is available. If the urgency is resolved, the procedure stops. Otherwise continue into the next step.

7) Activate the Subcontracting function. Estimate the quantity of each job for each subcontractor and modify the status of each job accordingly. If some urgency still exists, move the next step.

8) Activate the NEW-HIRE function. Identify the need for new hire by each operation type, project their skill growth, and verify the resources limits allowed for the total number of trainers.

9) Adjust related data and report the results.

2.8 Material Flow Simulation

Computer simulation is the most practical tool used in production planning and scheduling to validate static production plans in specific production environment [23,37]. Simulation also provides the user an effective tool to study potential bottlenecks and interact with it t improve the scheduled production. During the production stage, simulation can be used as a oduction control tool, when linked to an on-line shop floor control system to reflect up-to-minute work in inventory (WIP) status on the shop floor.

The main function of the proposed material flow simulation module is mainly for validation and further improvement of the static production plan. The flow chart is outlined in Figure 10. The major functions of the simulator include: a graphic layout generator, a graphic user interface, a simulation mechanism, and reporting subroutine. The graphic layout generator creates a graphic display of an apparel production system. The graphic display would be either a physical layout of workstations or a logical layout based on the operation sequence of a given garment. It would also show the current manning and inventory levels at each workstation. It would allow the user to dynamically assess the WIP buildup at each station and thus make timely decisions. The user interface function allows the production manager to specify the shop environment and modify decision rules and worker assignment. The simulation mechanism enumerates production activities as specified on the given schedule under a pre-set condition within resources constraints. The report function is responsible for summarizing simulated results and WIP inventory status. In response to the user's request, it may modify production plans and worker assignments currently stored in the database

2.9 Production Method Evaluation

The objective of the production method evaluation model is to evaluate the fitness of the bundle production method for a given job order and identify the best method based on minimal manufacturing cost. To do so, a cost model is proposed as below. For each production method (i.e., bundle, UPS, and modular), it considers setup cost, direct labor cost, indirect labor cost,

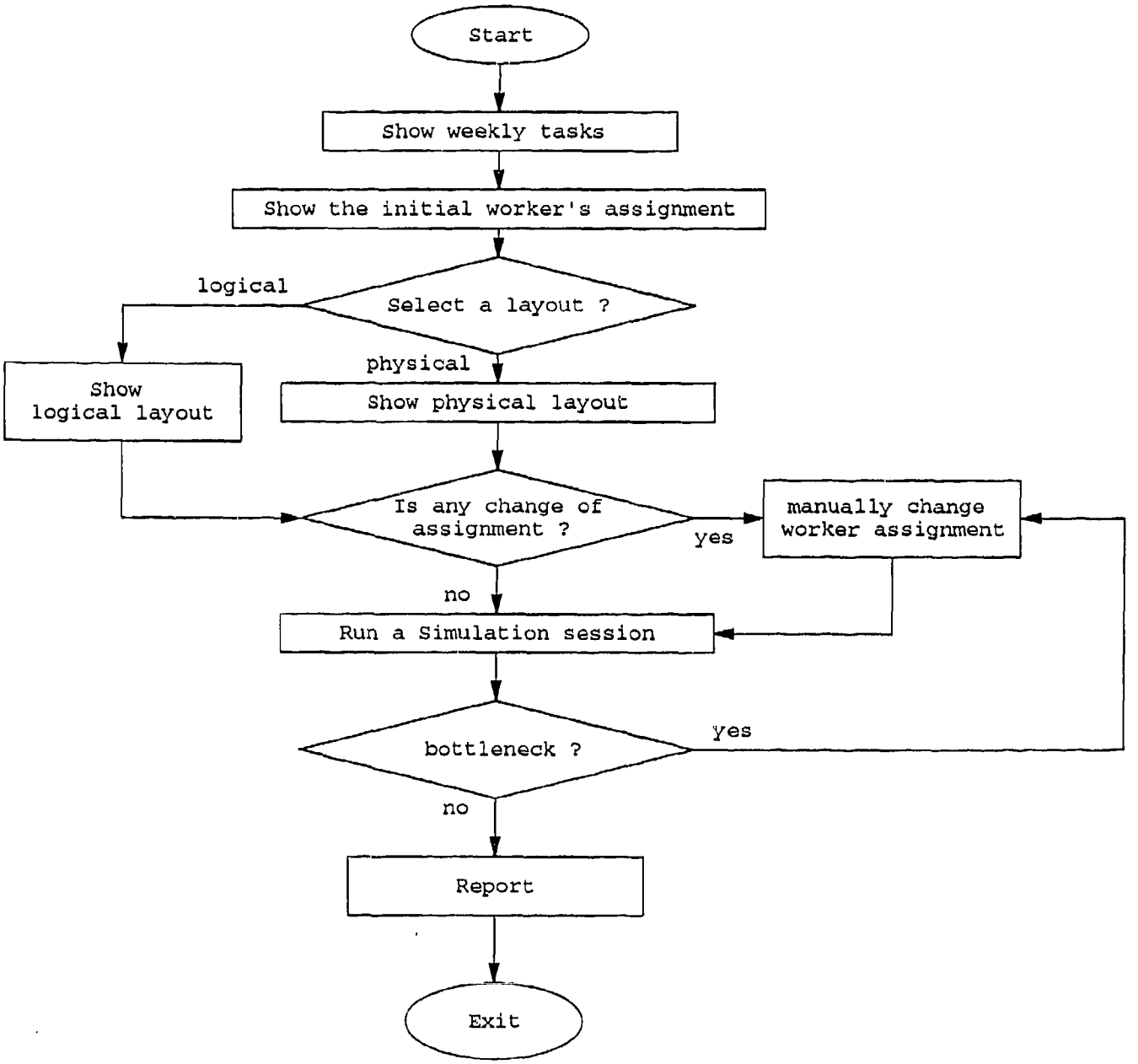


Figure 10 : Simulation Flow Chart

overhead cost, depreciation cost, and due date penalty. Because this model consists of four separable mathematical functions, the solution procedure is straight forward, although it includes some integer variables. However, the real difficulty shifts to the assessment of the cost factors. For example, the setup cost may not be easily available, especially for a unit production system. Another potential difficulty is the strategic considerations which may significantly affect the decision in selecting a production method for given job orders. For most apparel manufacturing firms, bundle production is the preferred method. Therefore the selection of the best production method in this study is considered advisory to the production manager.

$$\textit{Minimize:}\quad C_i = \left(\sum_{j=1}^{4} c_{ij}\right) \cdot Q + (T_i - D) \cdot c_p + c_{si} \qquad i = 1, 2, 3$$

where

C_i - total manufacturing cost for choosing production method i,

c_{i1} - direct labor cost for production method i,

c_{i2} - indirect labor cost for production method i,

c_{i3} - overhead cost for production method i,

c_{i4} - depreciation for production method i,

T_i - estimated finishing date for production method i,

D - due date,

c_p - unit penalty, and

c_{si} - setup (changeover) cost for production method i.

3. SYSTEM IMPLEMENTATION

This chapter summarizes the computer environment for and the implementation of the proposed system. One of the focus in this chapter is the implementation of the database system, which is critical to each intended engineering application.

3.1 System Environment

The proposed system is implemented in the DOS Windows environment. The Microsoft Windows/DOS 3.1 is used to support the Paradox/Windows 4.5 Database Management System. The Paradox Engine 3.0 is used as the data exchange tool. Turbo C++, version 3.0, is used to implement the application modules and integrate the system. The computer platform used in the implementation is a Dell/486 PC, which has a 4MB RAM. The minimum hard disk capacity is 30MB, because the Paradox/Windows 4.5 takes 20MB. The Paradox Engine occupies 4MB and the system program takes up 6MB. An EGA or VGA is required. Mouse is also a must.

As shown in Figure 11, the proposed system consists of a graphic user interface (GUI), a database management system, and a set of application modules. The GUI, which is implemented using the Paradox Application Language (PAL), provides the user with a friendly environment and a hierarchically structured pull-down menu. The database is accessed through *order, resource, knowledge*, and *schedule* menus. Each is provided with a full set of database operational functions including *add, delete, edit*, and *print* commands. The GUI allows the user to invoke any of the available application modules with a simple click on the mouse. Under each application, a hierarchy of menus are provided for the user to activate more specific commands within the application domain.

3.2 Database Design

The database is the heart of the proposed system. It supports all engineering application modules which frequently retrieve, update, and modify data residing in the database. PARADOX is selected to implement the proposed database system because of its flexibility and cost. It is a relational database and requires an entity-relationship data model to structure desired apparel production data. In the following three subsections, the design and implementation of the database system is described in the following design steps: conceptual, implementation, and physical design.

3.2.1 Conceptual design (entity-relationship data model)

The conceptual design draws up an architectural plan for the database system. It identifies entities required for the system, defines their attributes, and specifies the relationship between entities. The accuracy and completeness of a conceptual data model is critical to the database for meeting the need of the proposed system. An entity is defined as an object or event about which we choose to collect and store data. The entity-relationships for the proposed application are shown in Figure 12. In this figure, entities are represented by a rectangle. Diamonds are

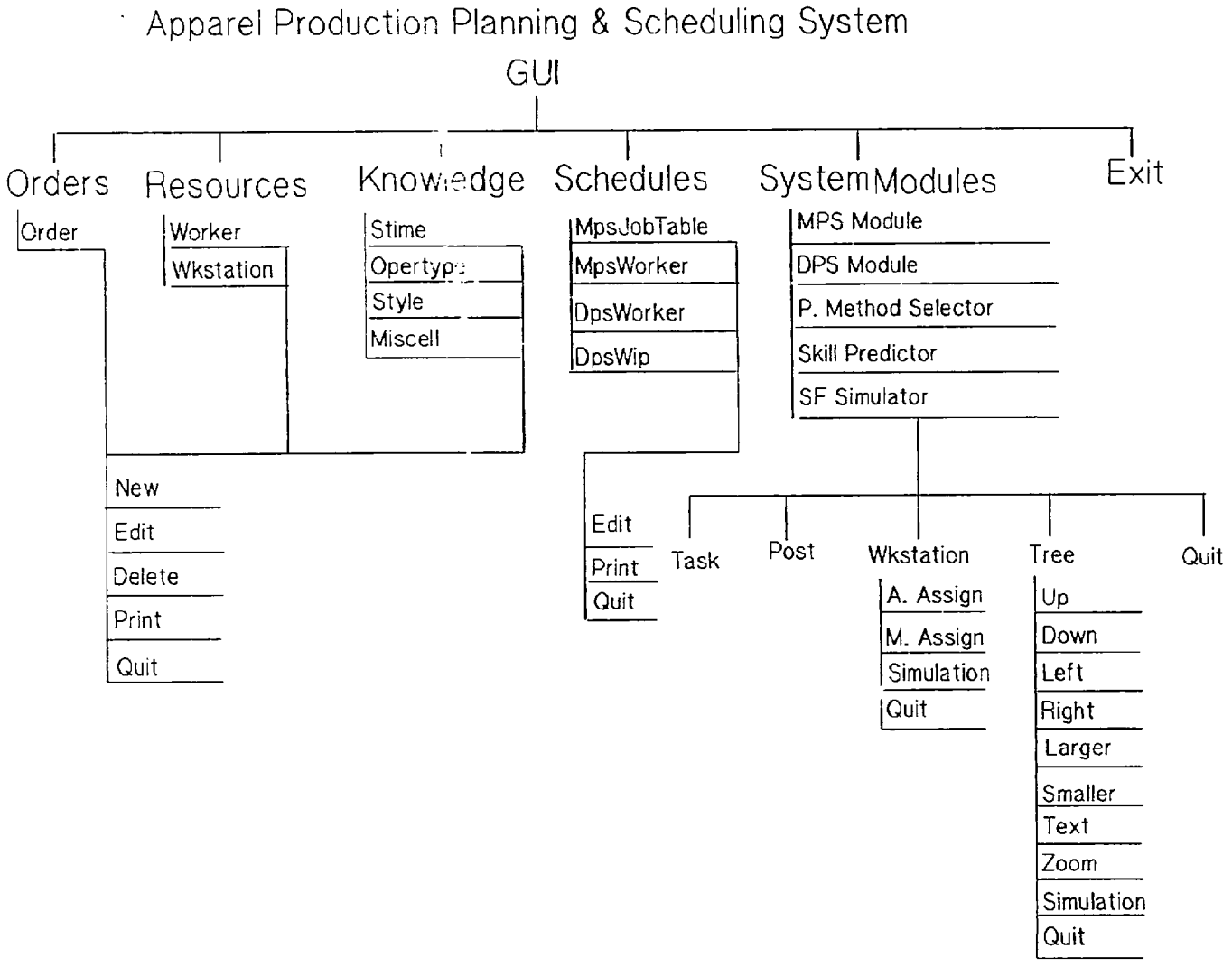


Figure 11: System Menu Hierarchy

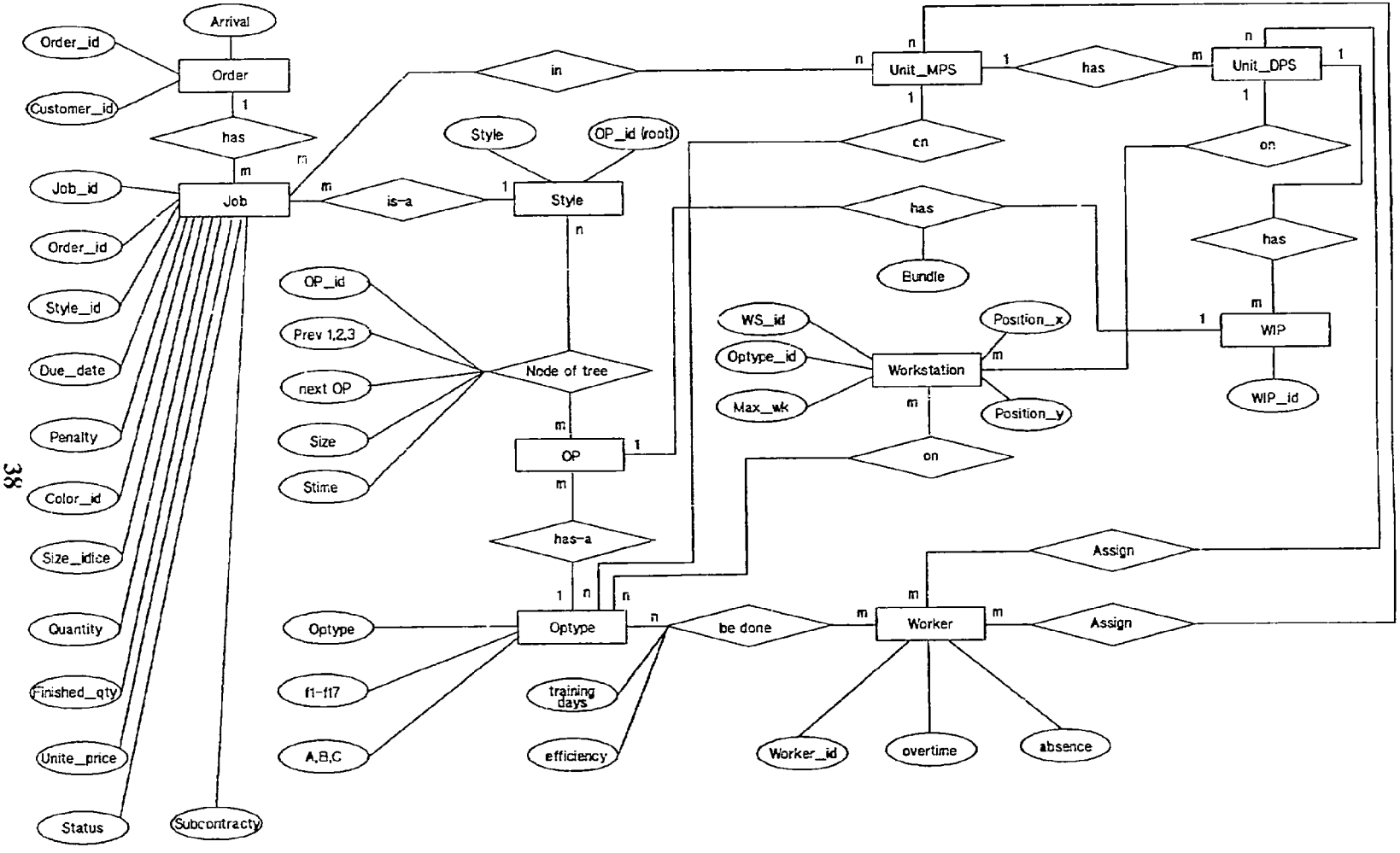


Figure 12 : Entity-Relationship Data Model

used to represent a relationship. A relationship between entities can be one to many (1→M) or multiple to many (M→N). Thus the characters on the lines between an entity (rectangle) and a relationship (diamond) specify the number of entities associated with each instance of the relationship. Both entity and relationship may have data element attributes (bubbles) associates with them. Data element bubbles for a relationship are considered intersection data. When an entity-relationship data model is implemented, diamonds with no associated data bubbles do not appear as a separate record type; but diamonds with bubbles require a distinct record type in implementation.

As outlined in Figure 12, a total of ten entities are defined to capture apparel design, orders, and production data for the intended application. They are orders, jobs, styles, operations, operation types, workstations, workers, unit master production schedule, unit detailed production schedule, and work in process (WIP) inventory. An apparel order, received from a customer, specifies one or many jobs. It is uniquely identified by its order ID and a customer ID. A job is a garment purchase item and is accompanied by a job ID, order ID, style ID, color, garment size, quantity, due date, penalty, and unit price. Each job has a pre-defined style, which is defined by a style ID and its root operation ID. It is represented by a tree of inter-connected nodes, of which each is related to an operation. Each node is also specified with an operation ID, its preceding operation IDs, subsequent operation ID, garment size, and standard processing time. Each operation belongs to an operation type, which is defined by an operation type ID and a set of characterization parameters. Workers are identified by an ID and associated with their overtime availability and scheduled absenteeism. A worker can perform one or multiple operation types at a different efficiency level depending on the number of training days. The assignment of each worker to an operation type is recorded in the master production schedule, whose immediate weekly production plan is subsequently decomposed into a detailed production schedule. The detailed plan further specifies the worker's assignment to a workstation and a job. Each detailed production plan specifies work in process inventories at station level, each identified by a WIP ID. Workstations are identified with an ID and specified with its capacity. Each workstation may handle multiple operation types but is assigned to only one operation type at any given time interval.

3.2.2 Implementation design (relational data model)

Implementation design is concerned with mapping the conceptual data model into a process logical model tailored for a selected database management system. The mapping process takes place in two stages. At the first stage, the conceptual model is translated into tables, of which each defines the attributes of a desired system entity. As a result, a total of fifteen tables are created for the proposed database system to capture orders (*ORDER*), jobs (*JOB*), operations (*OPER*), workers (*WORKER*), operation types (*OPTYPE*), worker to operation type relationship (*WK_OPTP*), workstations (*WS*), standard processing times (*STIME*), styles (*STYLE*), style operation relationship (*STYLE_OP*), unit master production schedule (*UMPS*), worker assignment in each unit master production schedule (*UMPS_WK*), worker to job assignment in detailed production schedule (*DPS_J_WK*), beginning and ending WIPs of each job on each day (*DPS_WIP*) at each workstation, and miscellaneous operation data (*MISCELL*). These tables will

be further detailed in the next section.

At the second stage, A set of "FORMs" are developed to organize entity data and their relationship retrieved from one or more tables. Through the use of Forms, the user can view and manipulate as much data as desired in a customized format. A total of ten forms are created in the Paradox Application Language to support the proposed system. They are: an order and job (*ORDER_JOB*) form, a worker and operation type (*WORKER_OPTYPE*) form, an operation type to job operation (*OPTYPE_OPERATION*) form, a standard processing time (*STIME*) form, a style to operation (*STYLE_OPERATION*) form, a miscellaneous (*MISCELL*) form, a unit MPS weekly production (*UMPS_THRY*) form, a unit MPS worker assignment (*UMPS_ASSIGNMENT*) form, a unit DPS worker assignment (*UDPS_ASSIGNMENT*) form, and a unit DPS daily WIP (*UDPS_WIP*) form.

Among the above ten forms, four of them are created with data retrieved from multiple tables. As shown in Figure 13, the *ORDER_JOB* form is created by linking the *ORDER* table to the *JOB* table with the key word *ORDER_ID*. In this form, the *ORDER* table is the parent table and *JOB* is the child table. This linkage has a 1→M relationship as one order may consist of one or more jobs. As shown in Figure 14, the *WORKER_OPTYPE* form is established by linking the *WORKER* table to the *OPTYPE* table with the key word *OPTYPE_ID*. The linkage has an M→N relationship. As shown in Figure 15, the *OPTYPE_OPERATION* form is created by linking the *OPTYPE* table to the *OPER* table with the key word *OPTYPE_ID*. In this form, the *OPTYPE* is called the parent and the *OPER* is called the child. The linkage has a 1→M relationship. On this form, all operations of the *OPTYPE* can be displayed and edited. The *STYLE_OPERATION* form is established with data from three tables. The *STYLE* table is 1→1 linked to the *OPER* table with the key word *OPERATION_ID*. Through this linkage, the operation type of the root operation of the style can be identified. The *STYLE* table is then linked to the *STYLE_OP* table with the key word *STYLE_ID* to create the form. The second linkage has a 1→M relationship, which is used to identify all operations of the style to be listed on the form. An example of this form is shown in Figure 16

3.2.3 Physical design (in Paradox 3.1)

Physical design is concerned with the structure of the record formats and selection of access methods, among others, to be used in the selected database system. The physical design of the above 15 tables in the Paradox are described in this section in the following four groups:

- job order data,
- product and production knowledge,
- manufacturing resource data, and
- existing schedules.

The job order tables capture both orders and individual jobs in each order. The product and production knowledge tables capture garment styles, operation structure and type, and standard sewing times, etc. The manufacturing resource tables store operator data, machine data,

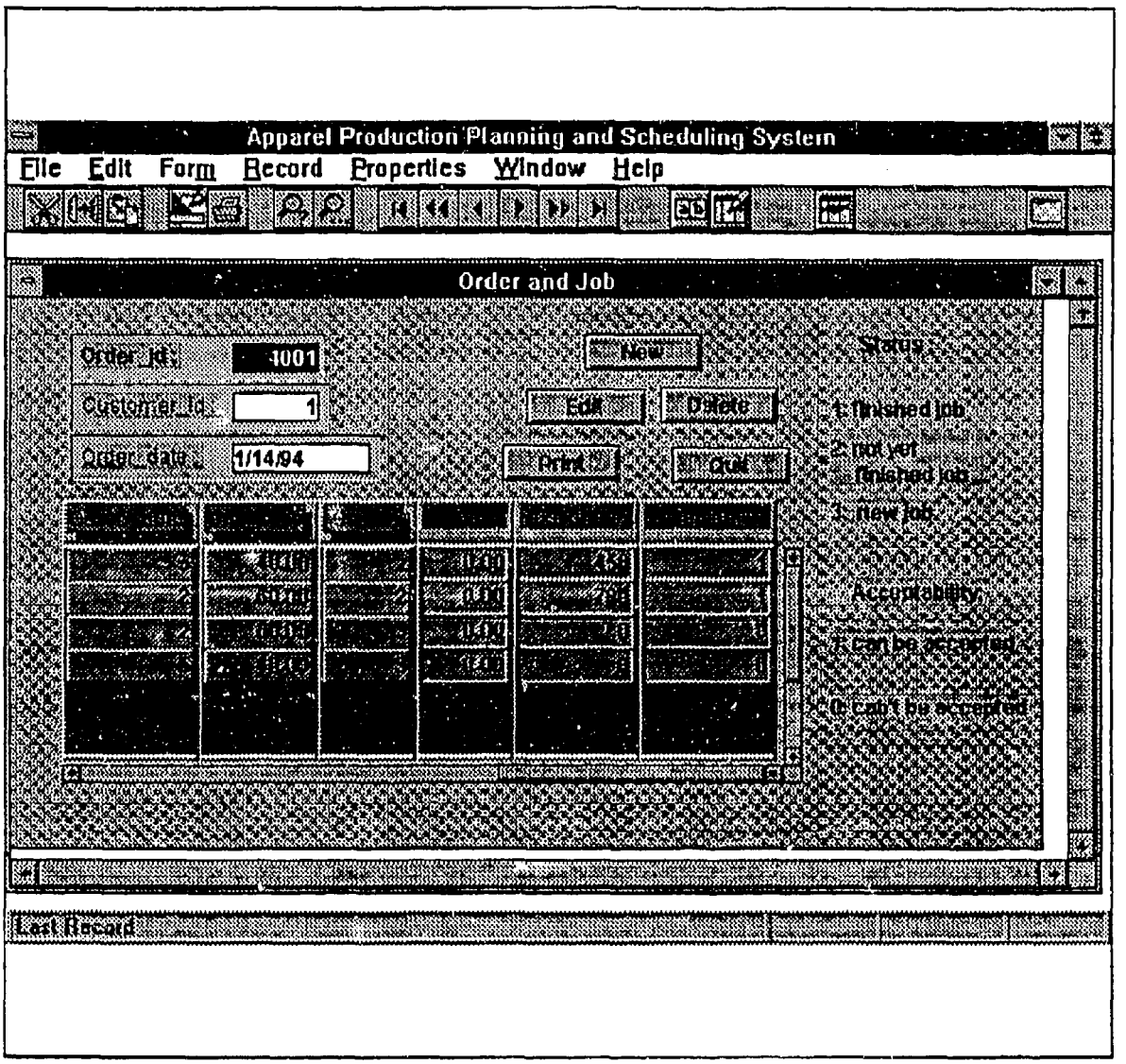


Figure 13 : The ORDER_JOB FORM

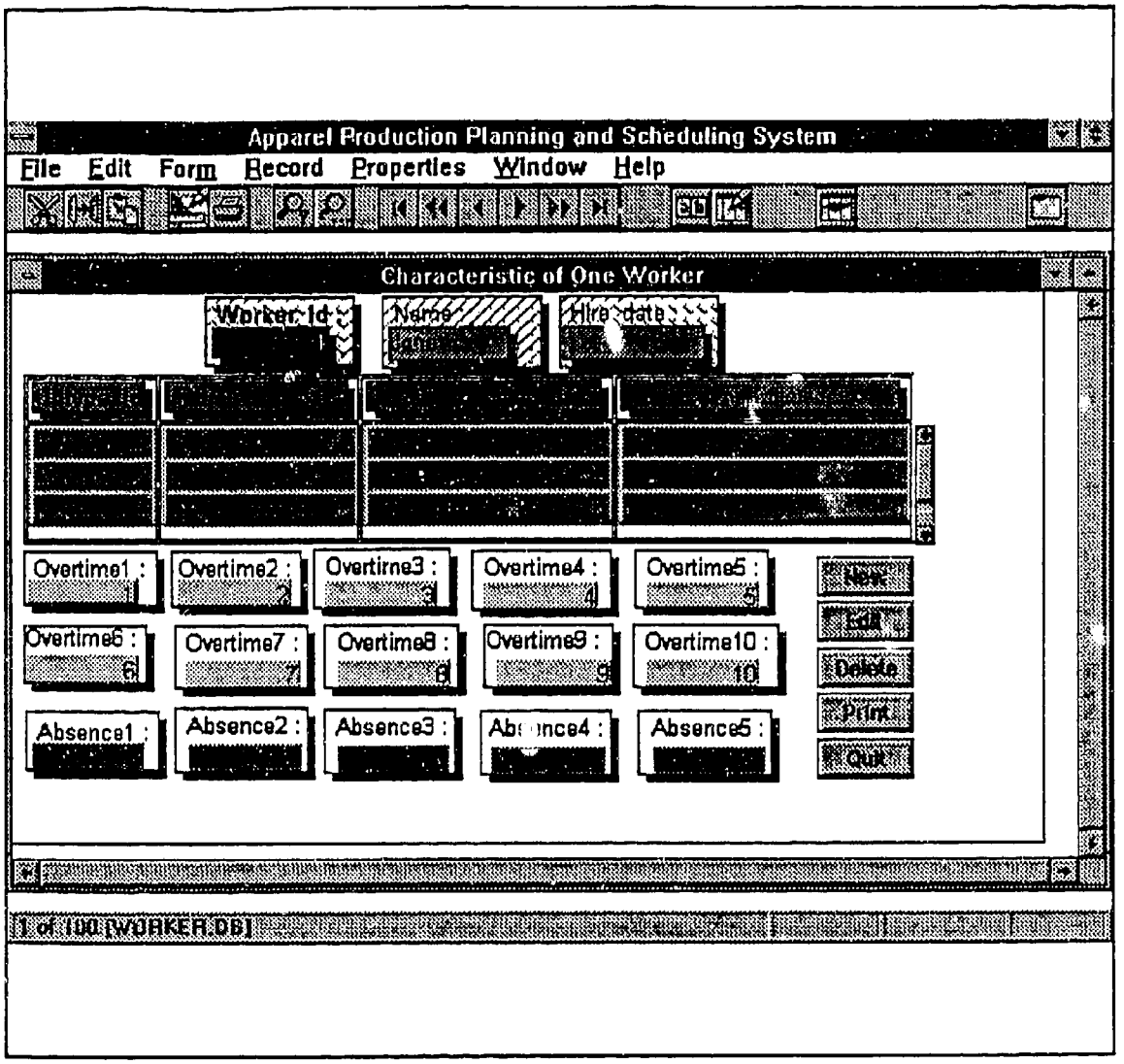


Figure 14 : The WORKER_OPTYPE FORM

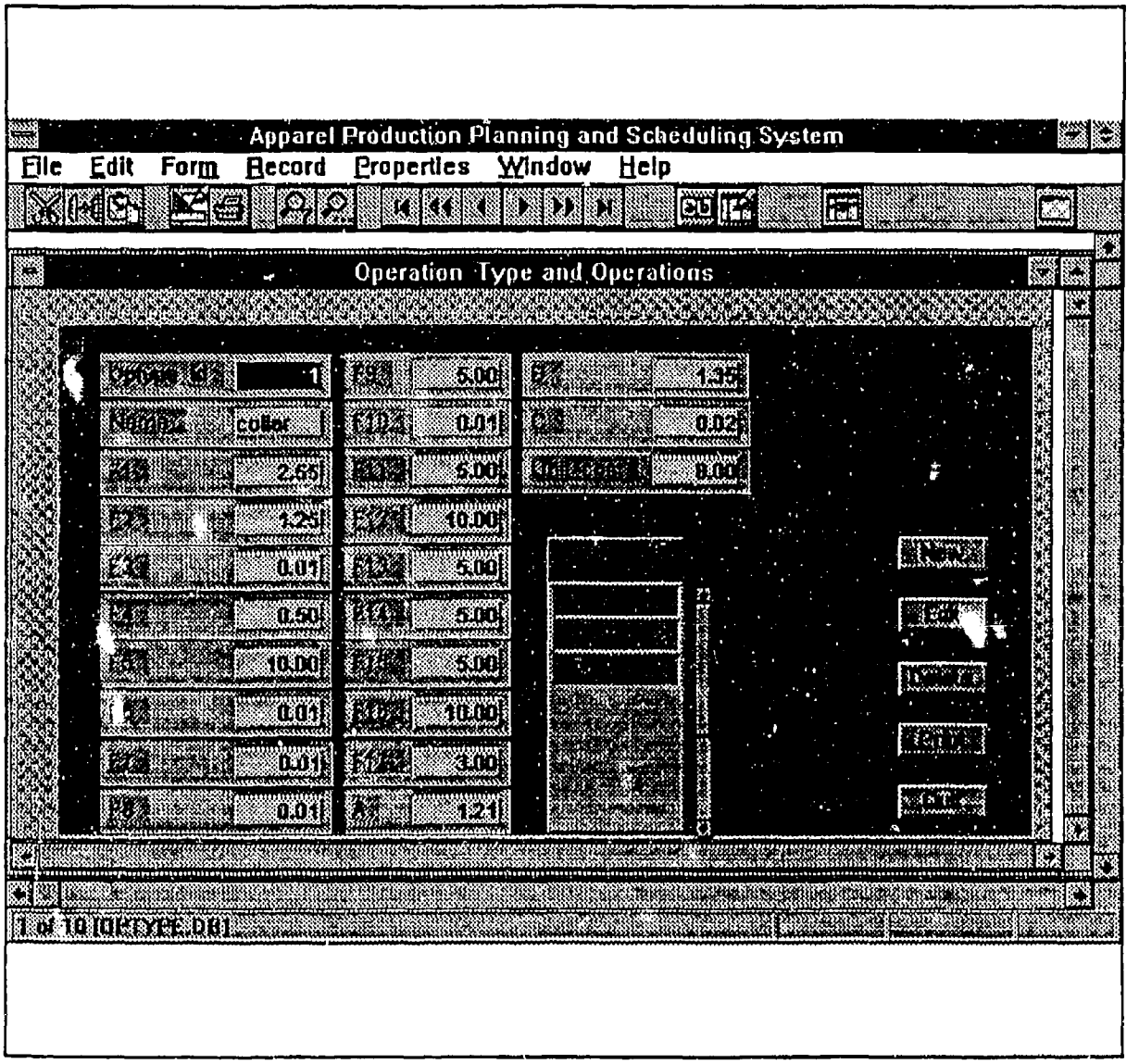


Figure 15 : The OPTYPE_OPERATION FORM

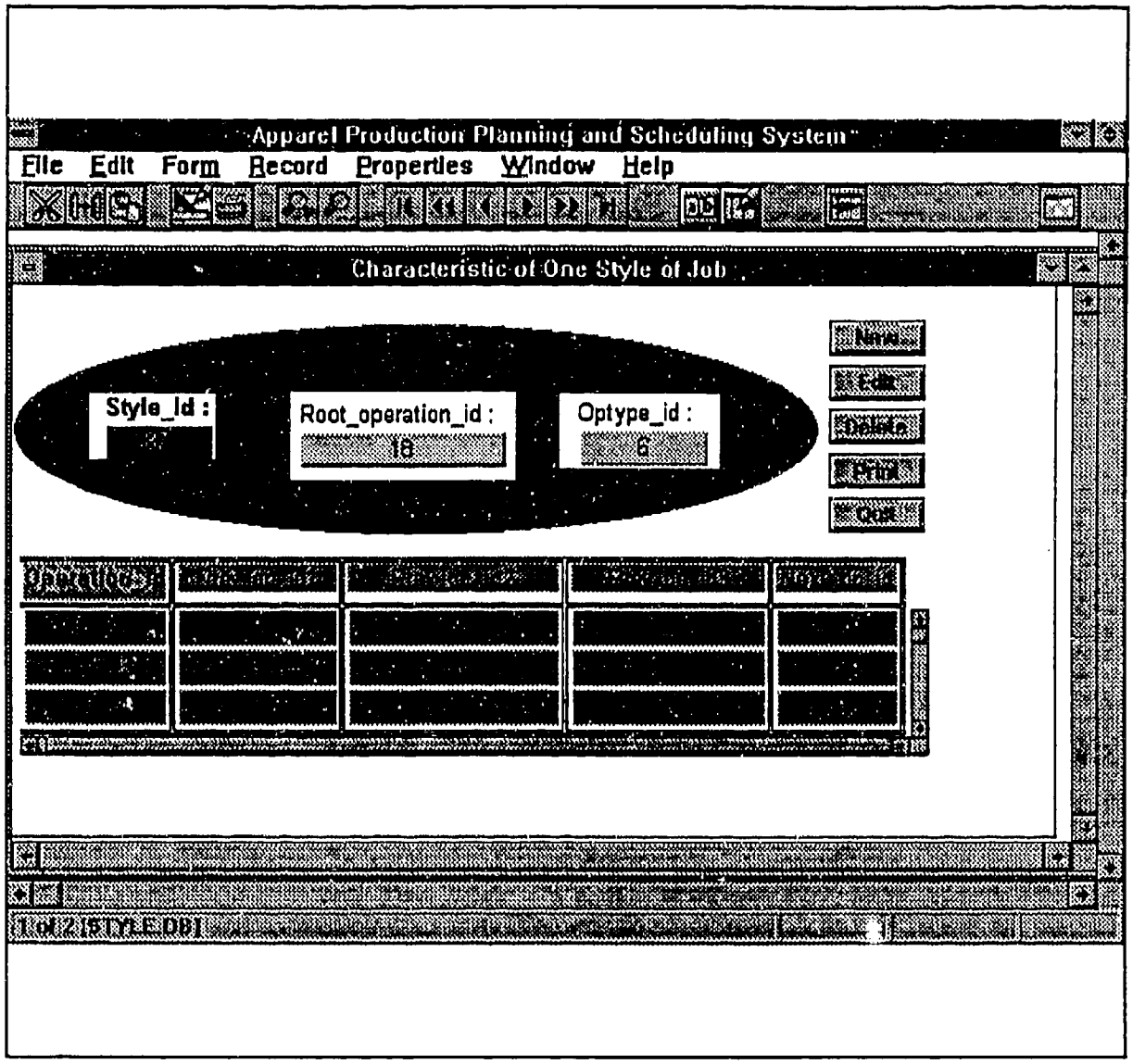


Figure 16 : The STYLE_OPERATIION FORM

and other miscellaneous data. The schedule tables store all process related decisions such as the master production schedules and detailed production plans. All data in the tables are integrated by key words when making special forms and reports. All data in the tables can be retrieved, added, edited, deleted, and printed, using the forms.

3.2.3.1 job and order tables

The data group contains two types of tables: *ORDER* and *JOB*. The *ORDER* table organizes order information such as order ID, customer ID, and ordering date. The above three attributes uniquely define an order. This table has no restrictions in the database.

As shown in Table 2, the *JOB* table captures all data about a job. The attributes of a job captured in the table include job ID, its parent order, the style ID, due date, penalty, unit price, cost, color ID, size ID, bundle size, status, acceptability, quantity, finished quantity, subcontractability, and subcontracted quantity. The style ID is used to specify the structure of operations to produce a garment of this job. The due date is usually set by customer at the point of ordering. The status attribute indicates whether the job is new, existing, or finished. The subcontractability specifies the maximum amount of subcontract allowed for the job. The subcontracted quantity attributes denotes the amount of work which are committed by a subcontractor.

The *JOB* table is built with restrictions such that each job has a unique job ID and is given by the system automatically. The due date must be within or later than the current week as specified in the miscellaneous table. The status and acceptability attributes cannot be modified by the user. The *JOB* table is updated by the master production scheduling module, usually done once a week.

3.2.3.2 manufacturing resource tables

This group includes three manufacturing resource related tables: operations (*OPER*), workers (*WORKER*), workers operation types (*WK_OPTP*), and workstations (*WS*). The *OPER* table defines each operation and its operation type for all the operations in the apparel firm. Regarding the restrictions imposed on this table, the operation ID is used as the key word for the table. The value of the key word is generated by the system and can only be changed by the super user. The operation type ID must be one of those defined in the *OPTYPE* table.

As shown in Table 3, the *WORKER* table captures each worker by the a set of attributes, which include worker ID, worker's name, hiring date, overtime availability and planned absence. In the current implementation, the overtime availability attribute allows a maximum of 10 days to be specified. The absence attributes allow capturing five days of planned absence. Each worker has an unique worker ID as the key, which is generated by the system. Other attributes can be added and edited by the user. The number of characters for worker's name must be less than eight characters. Both overtime and absence attributes are programmed to be retrieved only by the capacity planning module.

Table 2 : JOB Table

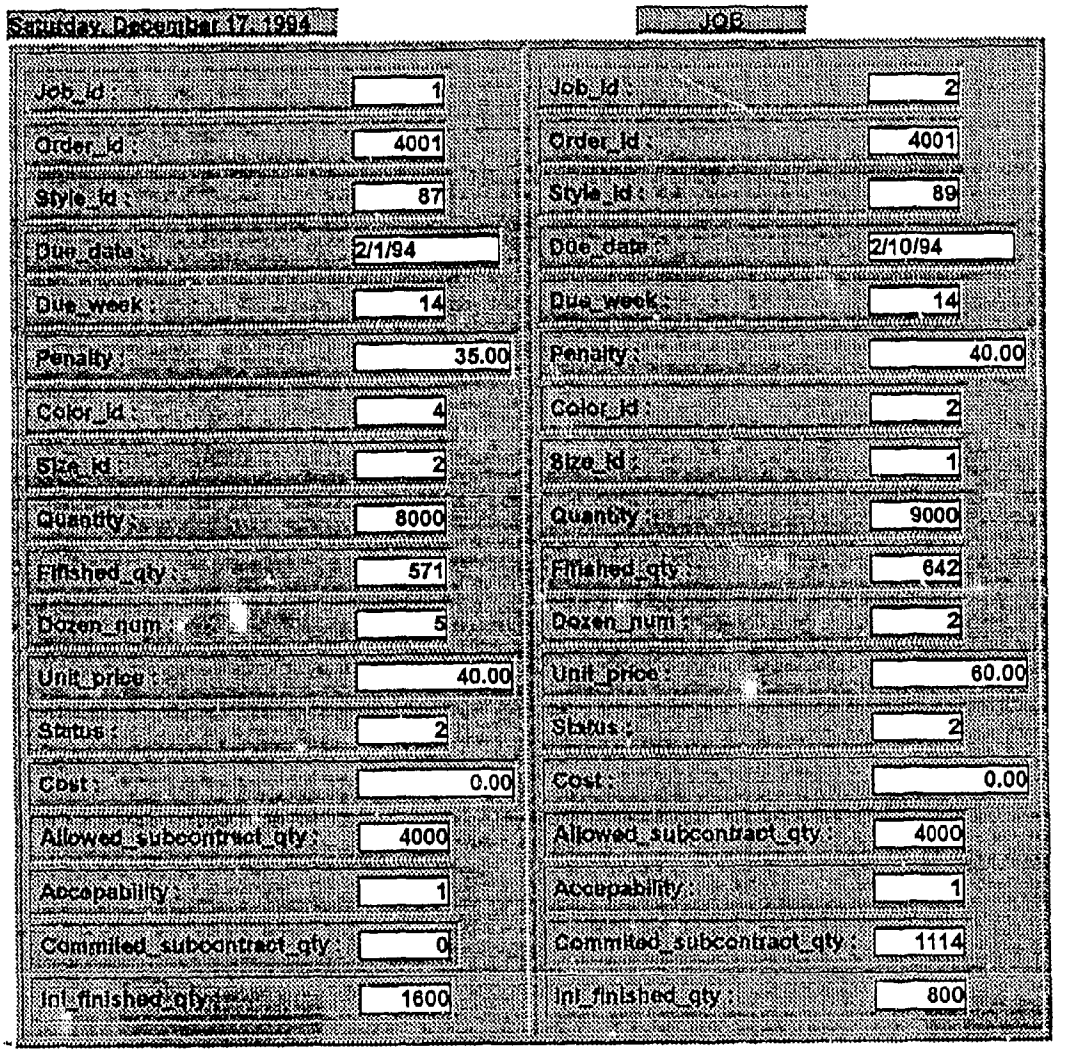
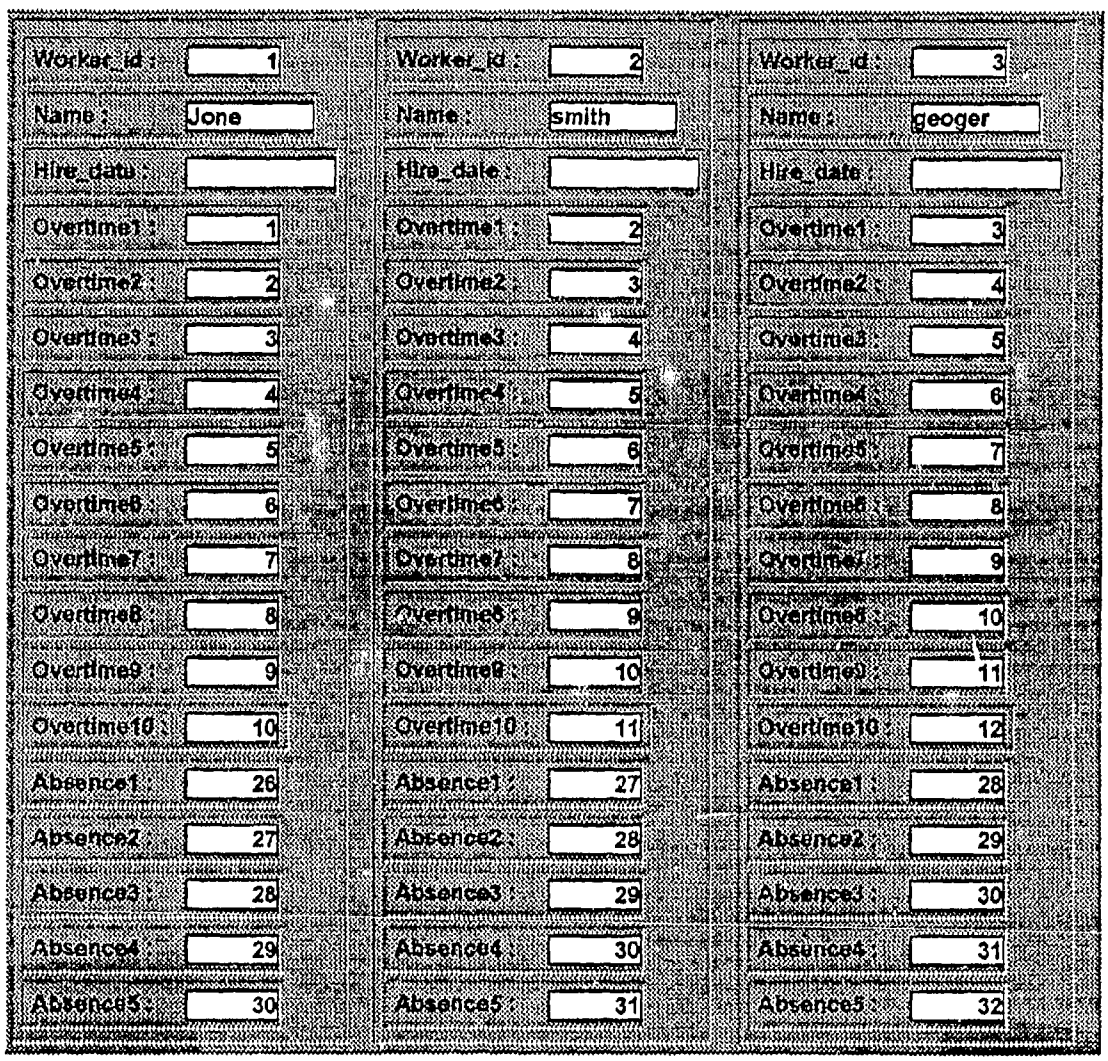


Table 3 : The WORKER Table

Sunday, December 18, 1994 WORKER



The *WK_OPTP* table is designed to capture the skill level of sewing workers at each operation type to support worker assignment decision. It captures worker's ID, operation type ID, initial skill level, number of days in training for the operation, current efficiency level, current training days, and the current training status. The skill level of a worker is assumed to be different at different operation type, depending on the amount of training and the nature of the operation type. On restrictions, the combination of worker ID and operation type ID is used as the key. All attributes except the current training status can be modified by the user. The training status can only be modified by the master production scheduling module and the material flow simulator, or by the super user.

As shown in Table 4, the *WS* table is designed to identify the attributes of workstations. These attributes includes workstation id, operation type ids which can be handled on that workstation, maximum number of workers, maximum inventory level, and location coordinates. The worker_id is defined as a key.

3.2.3.3 product and production knowledge tables

This group consists of the operation types (*OPTYPE*), standard time (*STIME*), style (*STYLE*), style operations (*STYLE_OP*), and miscellaneous (*MISCELL*) tables. As shown in Table 5, the *OPTYPE* table is designed to capture all operation types existing in an apparel firm. It contains the attributes of operation ID, operation name, the set of operation characterizing factors, and unit cost. For each operation type, a set of factors are captured in this table for sewing skill prediction. Non-sewing operations are assumed of having stable skill levels and therefore are not modeled in this study. The unit cost attributes stored the hour wage rate for a standard sewing worker at this operation type. This operation factors can be modified by the super user and are programmed to be retrieved only by the skill prediction module. The operation ID attribute is instantiated by the system. All other attributes in this table can be edited by the user.

As shown in Table 6, the *STIME* table is developed to capture standard processing time for each sewing operation. The standard time is determined by the garment style, size, and operation type. Therefore, this table stores standard processing time for each combination of all styles, sizes an operation types. Thus the combination of these three factors is used as the key of this table. The standard time can be retrieved by the master production scheduling, the detailed production scheduling, and the material flow simulation modules.

The *STYLE* table is created to store the style ID and its associated operation ID at root. This is needed because style is an attribute of a job. This attribute is designed to relate the style to the *STYLE* table such that the operation structure and sequence for the style can be retrieved. Once the root operation id was given, the whole operation structure and sequence can be found through use of the *STYLE_OP* table. The style ID attribute is used as the key, which is controlled by the system. The root operation ID can be added or edited by the user.

Table 4 : The WS Table

Saturday, April WS

Wkstation_id	Optype_id	Max_workers	Max_inventory	Position_x	Position_y
1	10	8	50		
2	2	8	50		
3	4	5	50		
4	7	8	50		
5	8	8	50		
6	1	8	50		
7	5	8	50		
8	6	10	50		
9	9	10	50		
10	4	5	50		
11	3	8	50		
12	1	10	50		
13	2	10	50		
14	3	10	50		
15	5	8	50		
16	5	8	50		
17	10	8	50		
18	6	8	50		
19	4	8	50		
20	1	8	50		
21	6	5	50		
22	8	5	50		
23	3	5	50		
24	1	8	50		
25	9	8	50		
26	10	8	50		
27	5	5	50		
28	6	8	50		
29	7	8	50		
30	9	8	50		

Table 5 : The OPTYPE Table

Saturday, December 17, 1994

OPTYPE

Optype_id :	1	Optype_id :	2	Optype_id :	3
Name :	collar	Name :	clew	Name :	placket
F1 :	2.55	F1 :	4.00	F1 :	6.80
F2 :	1.25	F2 :	1,025.00	F2 :	2.00
F3 :	0.01	F3 :	1.00	F3 :	0.01
F4 :	0.50	F4 :	2.00	F4 :	2.50
F5 :	10.00	F5 :	10.00	F5 :	10.00
F6 :	0.01	F6 :	0.01	F6 :	0.01
F7 :	0.01	F7 :	0.01	F7 :	3.00
F8 :	0.01	F8 :	0.01	F8 :	3.00
F9 :	5.00	F9 :	0.01	F9 :	0.01
F10 :	0.01	F10 :	0.01	F10 :	0.01
F11 :	5.00	F11 :	5.00	F11 :	10.00
F12 :	10.00	F12 :	5.00	F12 :	10.00
F13 :	5.00	F13 :	5.00	F13 :	5.00
F14 :	5.00	F14 :	5.00	F14 :	10.00
F15 :	5.00	F15 :	5.00	F15 :	5.00
F16 :	10.00	F16 :	3.00	F16 :	10.00
F17 :	3.00	F17 :	3.00	F17 :	3.00
A :	1.21	A :	0.92	A :	1.13
B :	1.35	B :	1.49	B :	1.44
C :	0.02	C :	0.03	C :	0.03
Unit_cost :	8.00	Unit_cost :	8.00	Unit_cost :	7.00

Table 6 : The STIME Table

Style_id	Size	Op_id	Standard_time
87	1	1	1224
87	1	2	520
87	1	3	326
87	1	4	816
87	1	5	1108
87	1	6	804
87	1	7	1108
87	1	8	1164
87	1	9	1980
87	1	10	311
87	1	11	1014
87	1	12	1836
87	1	13	981
87	1	14	806
87	1	15	727
87	1	16	704
87	1	17	1176
87	1	18	1176
87	2	1	1020
87	2	2	441
87	2	3	271
87	2	4	680
87	2	5	973
87	2	6	870
87	2	7	823
87	2	8	970
87	2	9	1650
87	2	10	259
87	2	11	645
87	2	12	1530
87	2	13	826
87	2	14	505
87	2	15	606
87	2	16	557
87	2	17	980
87	2	18	980
88	1	2	1290
88	1	8	647
88	1	13	1280
88	1	14	1260
88	1	19	3000
88	1	20	865
88	1	21	704
88	1	22	1510
88	1	23	757
88	1	24	1580
88	1	25	771
88	1	26	1160
88	1	27	580
88	1	28	2330
88	1	29	1860
88	1	30	1430
88	2	2	1419
88	2	8	712
88	2	13	1408

As shown in Table 7, the *STYLE_OP* table is created to capture the entire operation structure of each garment style with six attributes. The style ID attribute is used as the key in combination with the operation ID to the right *STYLE_OP* entity instance. Three attributes are used to capture up to three preceding operations. The fourth attribute is used to capture its next operation ID. By chaining the interrelationship of the operations within the style, an entire operation tree can be revealed. The capture of three preceding operations implies that an apparel assembly operation in this implementation can assemble up to three garment pieces. The value of all attributes can be modified by the user. These data are retrieved and used by both the master production scheduling and the detailed production planning modules.

The *MISCELL* table, as shown in Table 8, is designed to capture miscellaneous data which do not belong to other entities but are required by the system. It specifies four attributes: application level, current week, number of days in a week, and total daily working hours available. The application level is set at two levels: testing and application. The system works in different manner under the different levels. The current week attribute records the current week in relation to the week of system initialization. The number of days per week attribute affect the total available working hours in each week. It could be different, although normally an apparel firm operates five days a week. Similarly the working hours per day captures the total working hours each day. It usually has an 8-hour shift. If it runs two shifts, the number of hours doubles. The allowance for overtime is affected by this attributes. No key is used in this table. The application level attribute assumes 0 or 1. The master production scheduler can change the current week attribute when the status is 1. When it is switched to 0, all attributes in the table can be modified by the user.

3.2.3.4 production schedule tables

This group includes four tables: the unit master production schedule (*UMPS*), worker assignment for each *UMPS*, detailed production schedule per job and worker (*DPS_J_WK*), and its associated work in process inventory (*DPS_WIP*).

The *UMPS* table, as shown in Table 9, stores the master production schedule, which indicates the scheduled throughput of each job in each week in the planning horizon. The combination of the week ID and job ID attributes is the key word. All attributes are updated by master production scheduler and can be retrieved by the user and detailed production scheduler.

The *UMPS_WK* table, as shown in Table 10, is designed to store the assignment to an operation type during master production scheduling. The assignment of workers to a specific job and work station is not yet available at this time. The key word is the combination of its three attributes: week ID, worker ID, and *OPTYPE* ID. All attributes can be updated by the master production scheduler but cannot be altered by the user or other modules.

As shown in Table 11, the *DPS_J_WK* table stores the detailed production schedule, which are identified with worker ID, operation type ID, workstation ID, job ID, and the expected daily throughput by the worker. The key word is the combination of its attributes: the date,

Table 7 : The STYLE_OP Table

Saturday, December 17, 1994 STYLE_OP

Style_id	Operation_id	Prev_op_id1	Prev_op_id2	Prev_op_id3	Next_op_id
87	1	-1	0	0	2
87	2	1	0	0	3
87	3	2	0	0	4
87	4	3	0	0	5
87	5	4	0	0	9
87	6	-1	0	0	7
87	7	6	0	0	8
87	8	7	0	0	9
87	9	8	5	0	10
87	10	9	0	0	11
87	11	10	0	0	15
87	12	-1	0	0	13
87	13	12	0	0	14
87	14	13	0	0	15
87	15	11	14	0	16
87	16	15	0	0	17
87	17	16	0	0	18
87	18	17	0	0	-2
89	2	19	0	0	20
89	8	23	0	0	24
89	13	25	0	0	14
89	14	13	0	0	26
89	19	-1	0	0	2
89	20	2	0	0	21
89	21	20	0	0	22
89	22	21	0	0	26
89	23	1	0	0	7
89	24	8	0	0	27
89	25	-1	0	0	13
89	26	14	0	0	27
89	27	24	26	0	25
89	28	22	27	0	29
89	29	25	0	0	30
89	30	29	0	0	-2

Table 8 : The MISCELL Table

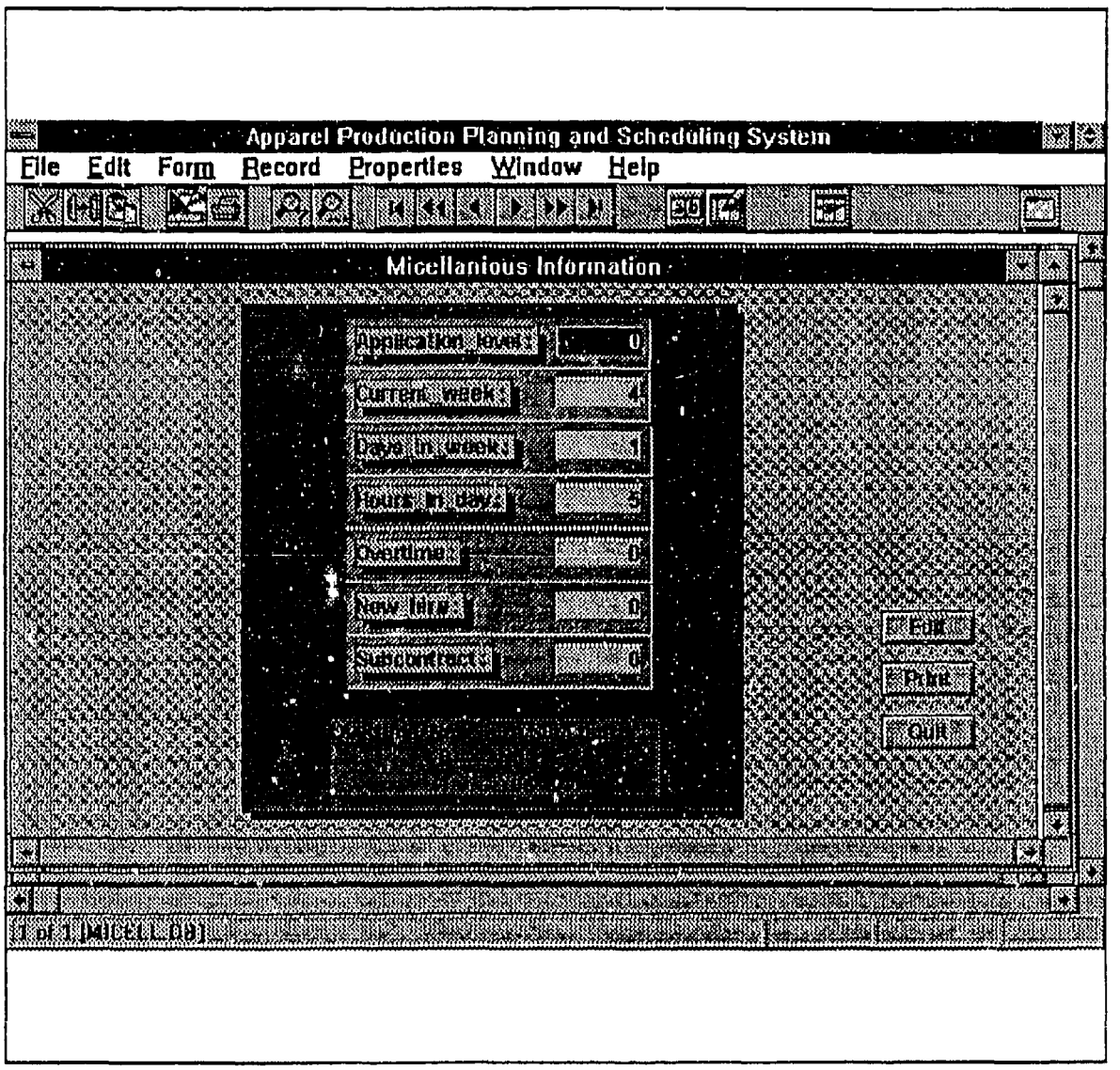


Table 9 : The UMPS Table

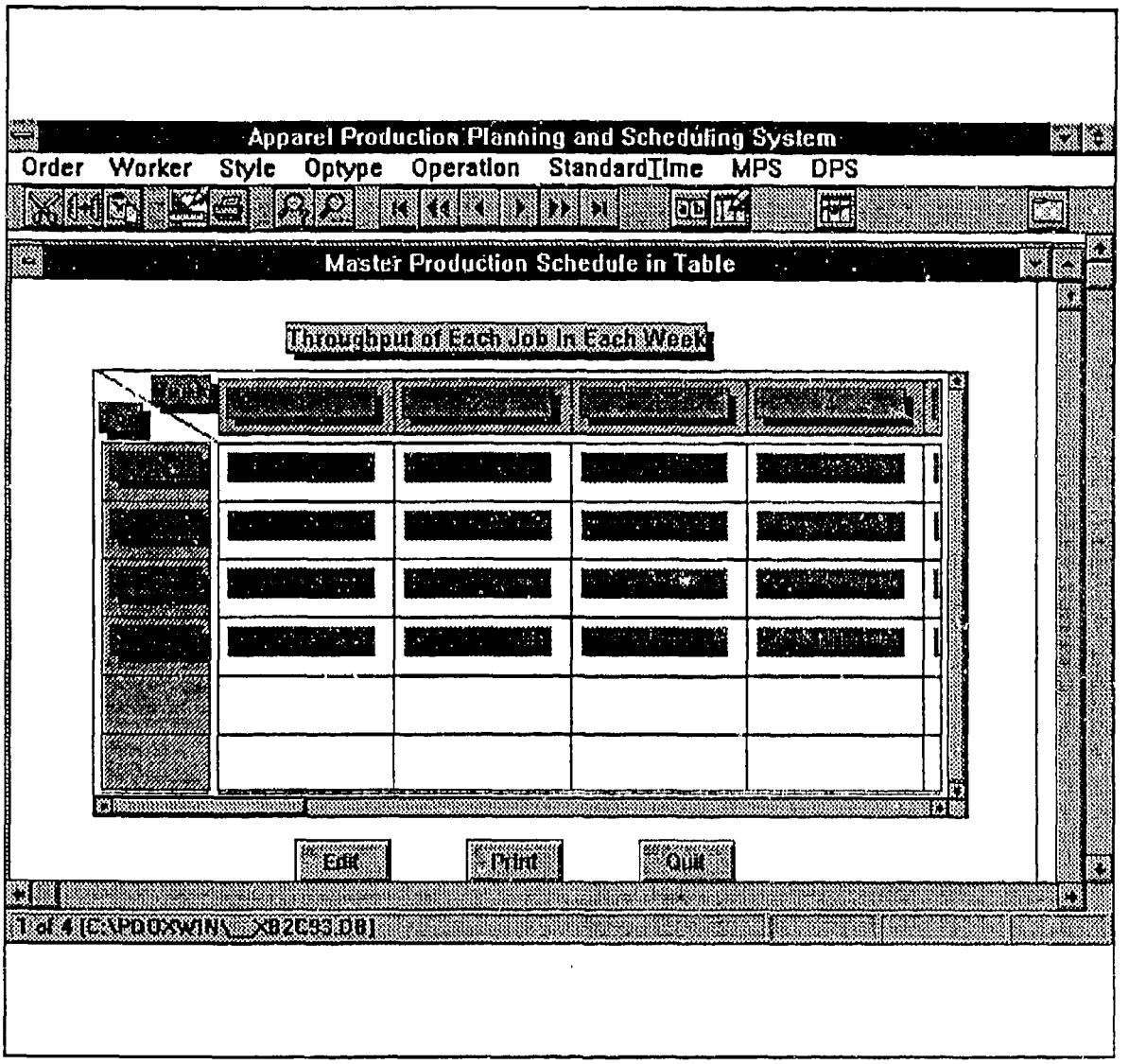


Table 10 : The UMPS_WK Table

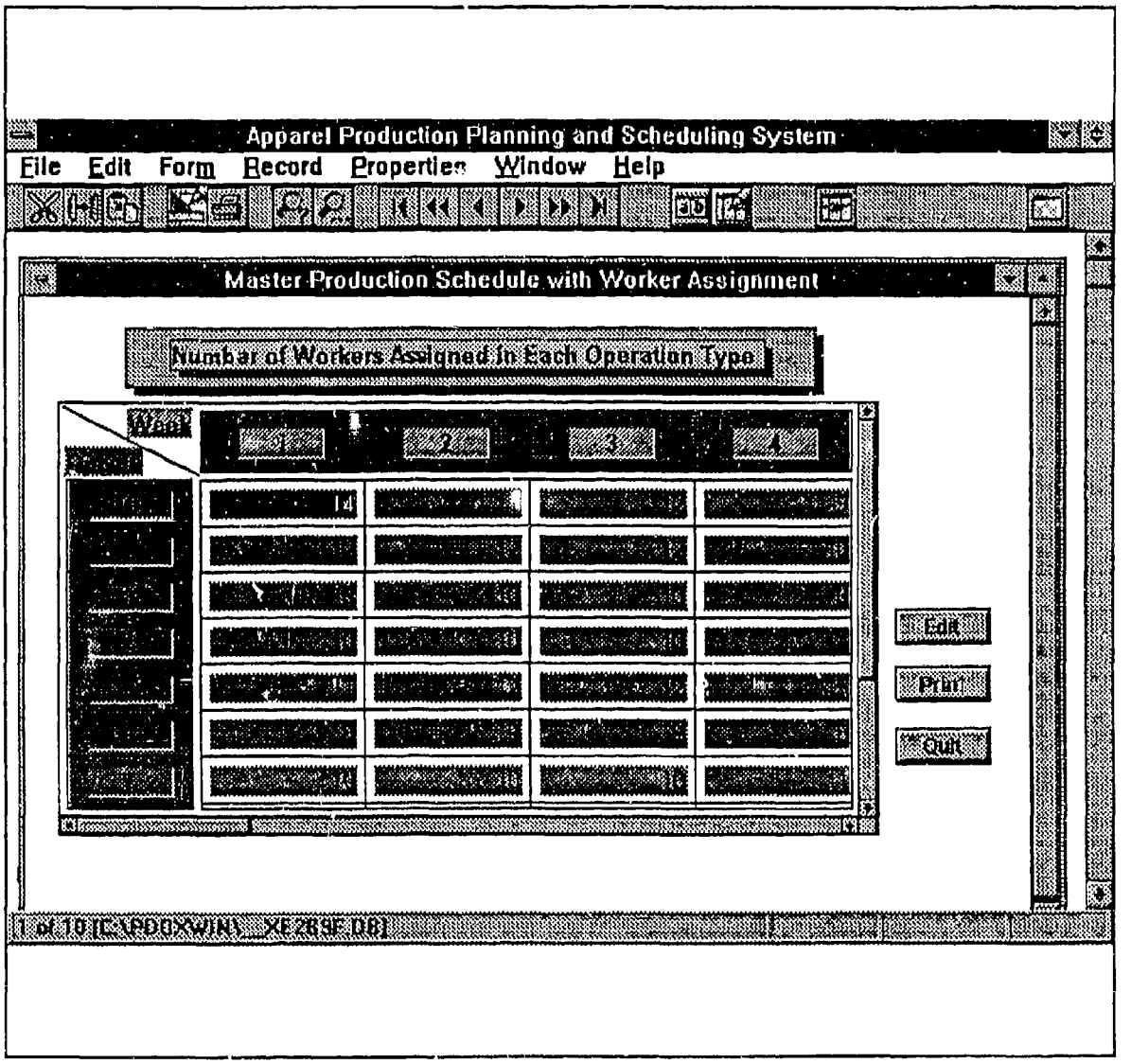
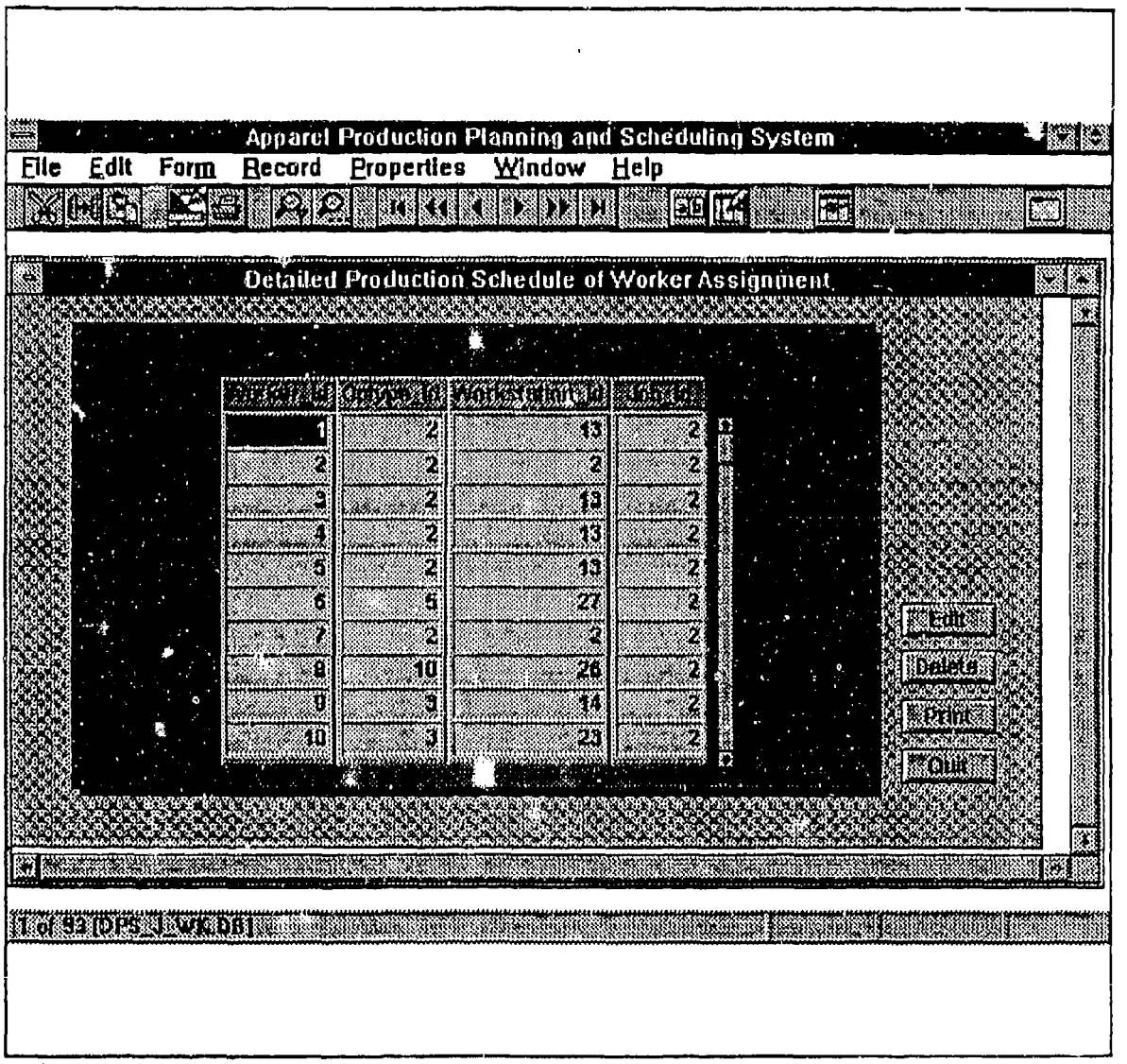


Table 11 : The DPS_J_WK Table



week, *OPTYPE* ID, workstation ID, and job ID. All attributes are updated by the detailed production scheduler and the material flow simulator. The user and the simulator can view and edit the throughput data.

The *DPS_WIP* table, shown in Table 12, is designed to capture an additional portion of the detailed production schedule data, which records the total work in process inventory including daily input, output, and the current inventory level of each job at each workstation. The key word of this table is the combination of the date, week, and job ID attributes. All attributes are updated by the detailed production scheduler and the simulator. In the current implementation, the system user is allowed to update these inventory data. Ideally the proposed system should be linked to a shop floor data collection system and thus WIP data could be on-line updated with data collected from the shop floor system.

3.3 Skill Prediction Module

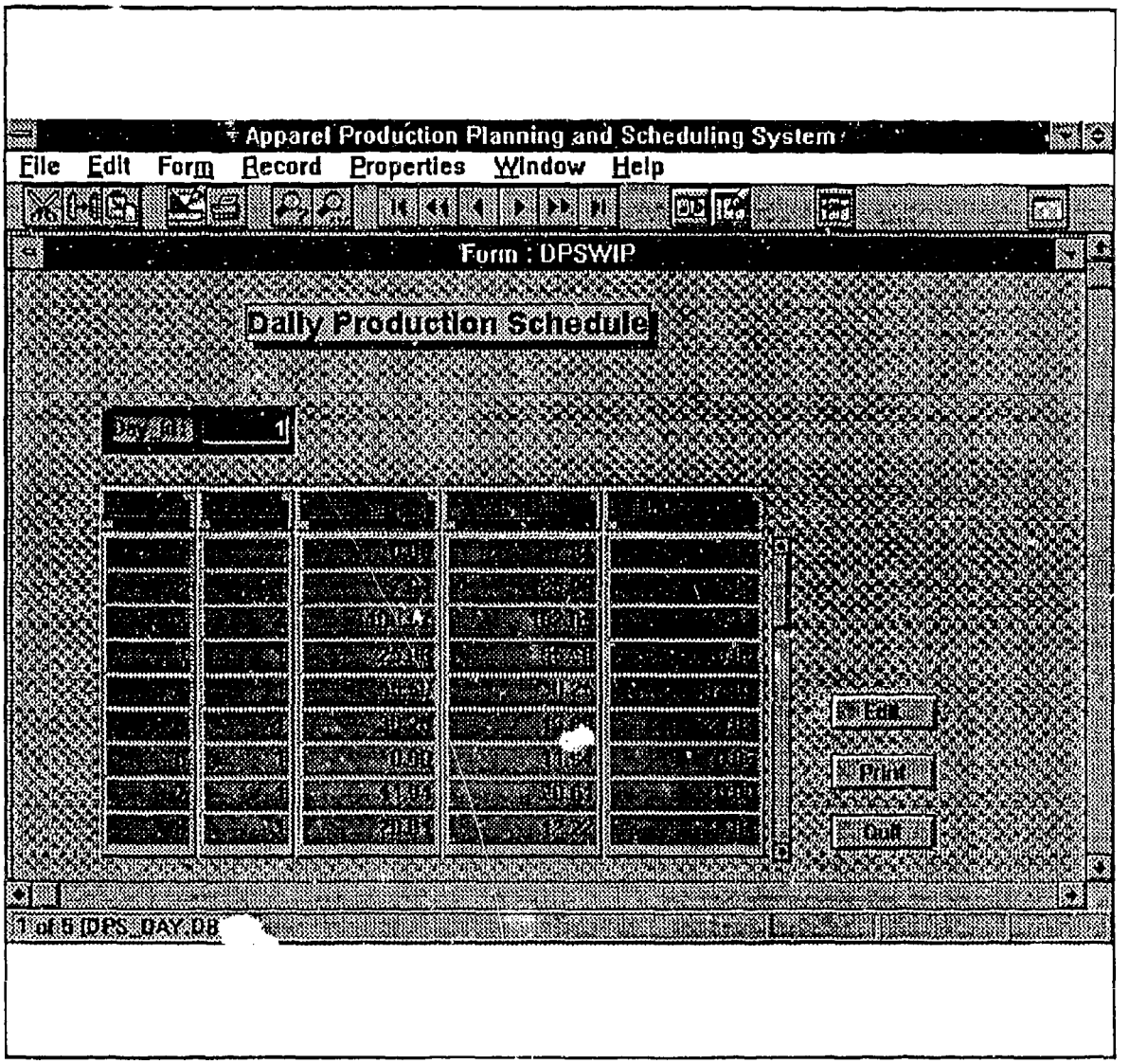
The sewing skill prediction module is implemented according to the skill prediction model described in the previous chapter. The main program for this module is called *NonLinearFun*. The module interacts with the database for data from two tables: *WK_OPTP* and *OP_TYPE*. From the *WK_OPTP* table, the following data are retrieved: days of initial training and days of current training. From the *OP_TYPE* table, the operation-related parameters for the selected operation are retrieved, which include the 17 independent variables and the 3 predictors (A, B., and C). The output data are used to update the current training days and current efficiency stored in the *WK_OPTP* table.

Although the module is designed for on-line use by other system application modules, this module can also be used off-line to generate various learning curves for other purposes. The user activates the module by selecting the submenu *Sewing_Skill_Predictor* in the *System Modules* menu and then specifying a desired operation type. After the ID for the desired operation type is entered, the learning curve is generated, which can be displayed and printed out as shown in Figure 17. The user can also get detailed skill efficiency on any training day from the curve.

3.4 Production Method Evaluation Module

The implementation of the production method evaluation module is based on the evaluation model described in chapter two. The module interacts with the database for various cost related data such as job ID, style ID, size ID, due dates, penalty, unit cost, depreciation, overhead, standard time, and setup. The output data includes the preferred production method and cost figures. These output data serve as only a point of interest to the manager and are not stored in the database because no future use is anticipated. Due to the fact that most apparel firms do not have the various production systems and the bundle system is our research focus in this study, this result from this module is only for the user's reference and should not affect other scheduling functions.

Table 12 : The DPS_WIP Table



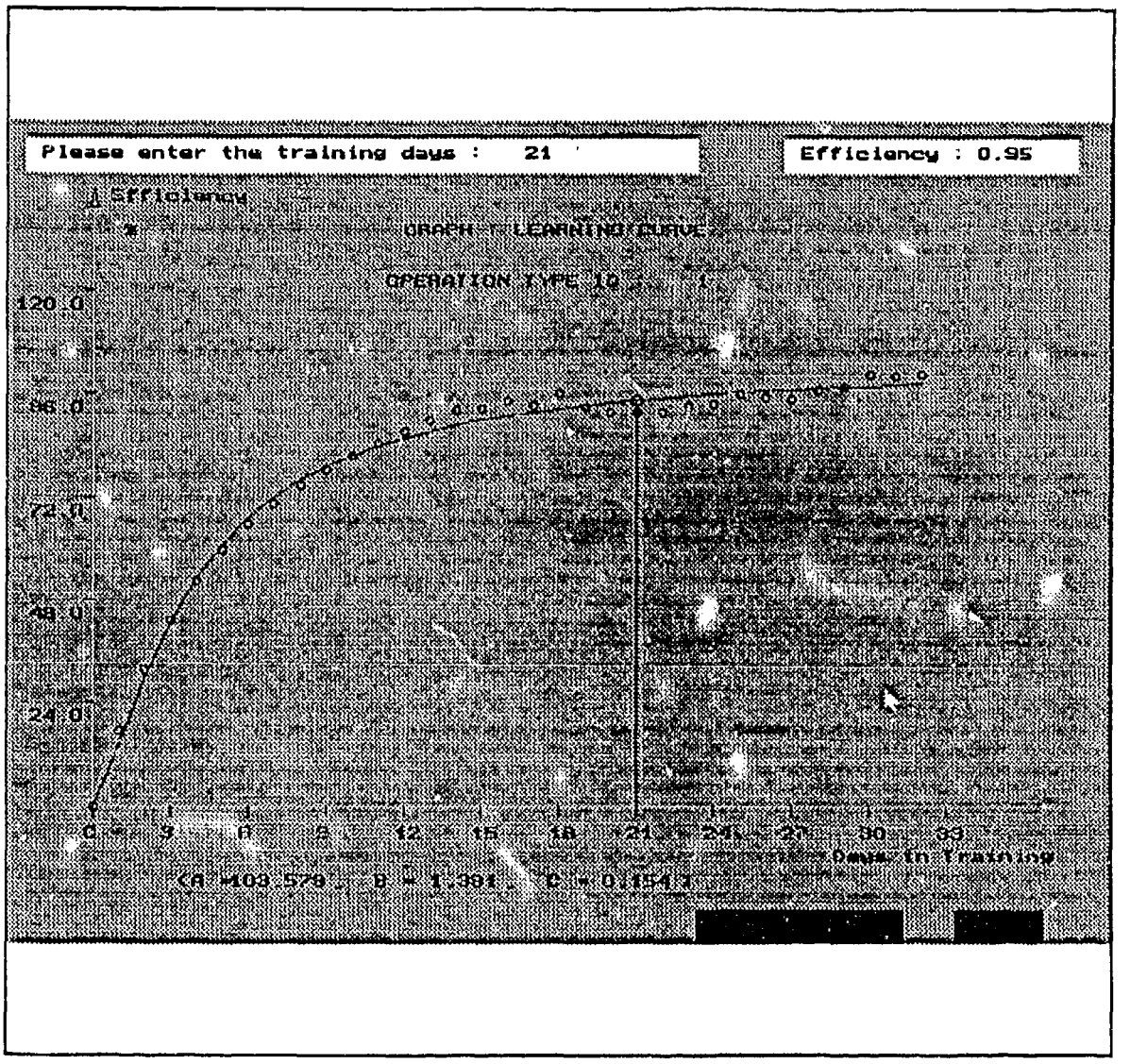


Figure 17 : A Projected Learning Curve

3.5 Master Production Scheduling Module

The master schedule module is implemented according to the MPS solution procedure proposed in Chapter 2. This module is a core of the proposed system. It consists of 18 program routines and interacts with the database to retrieve data from eight tables and write to five tables. From the *MISCELL* table, it acquires current week, days per week, and work hours per day. From the JOB table, it retrieves job ID, style ID, due dates, penalty, size ID, total quantity, and bundle sizes. From other tables, it obtains more data regarding workers, operation types and garment structures. The major output is an updated master production schedule. It also updates projected job and worker status, among others.

The first program routine is called *CommittedSubQty*, which writes an integer value to *JOB* table. The second routine is called *FinishedQty*, which records projected finished quantity in the JOB table. This function searches through each job to update the beginning finished quantity and ending finished quantity for each week, according to the projection. The *GetAllOptype* is the third function which retrieves all operation types from the *OPTYPE* table. The function opens the *OPTYPE* table and retrieves the value from all the related records in the database. The fourth function is called *GetAllWorker*, which retrieves all available workers from the *WORKER* table. It opens the *WORKER* table and retrieves all the records from the database. The C++ class definition for workers in this implementation is partially shown as below:

```
class Worker{
        private:
        int     worker_id;
        int     training_days[MAX_NUM_OpTYPE];
        int     last_optype_id;
        int     curr_optype_id;
        int     overt[MAX_NUM_OVERTIME];
        int     abse[MAX_NUM_ABSE];
        int     avail;
        public:
        Worker(int a_worker_id);
        Worker();
        ~Worker();
        void    SetId(int a_worker_id);
        int     GetId();
        void    SetTrainningDays(int a_optype, int a_days);
        int     GetTrainningDays(int a_optype);
        void    SetLastOptypeId(int a_last_optype_id);
        int     GetLastOptypeId();
        void    SetCurrOptypeId(int a_curr_optype_id);
        int     GetCurrOptypeId();
        void    SetOvert(int a_index, int a_week);
        int     GetOvert(int a_index);
```

```
void    SetAbse(int a_index, int a_week);
int     GetAbse(int a_index);
void    SetAvail(int a_avail);
int     GetAvail();
double          OptypeEff(int optype_id, Optype optype[]);
};
```

The fifth MPS function is called *GetCurrWeek*, which retrieves the current week datum from the *MISCELL* table. The next function is *GetJob*, which retrieves all related job data from the *JOB* table. It opens the *JOB* table, searches for jobs which have a matched job ID and returns values of its style ID, size ID, due dates, quantity, finished quantity, subcontracted quantity and bundle size. The job class is defined as below.

```
class Job{
        private:
        int     job_id;
        Style*  style;
        int     due_week;
        int     size_id;
        int     quantity;
        int     finished_qty;
        int     sub_qty;
        int     dozen_num;
        int     status;
        public:
        Job();
        ~Job();
        void    SetJobId(int a_job_id);
        int     GetJobId();
        void    SetStyle(Style*);
        void    GetStyle(Style*&);
        void    SetDueWeek(int a_due_week);
        int     GetDueWeek();
        void    SetSizeId(int a_style_id);
        int     GetSizeId();
        void    SetQty(int a_quantity);
        int     GetQty();
        void    SetFinishedQty(int a_finished_qty);
        int     GetFinishe  ty();
        void    SetSubQty(int a_sub_qty);
        int     GetSubQty();
        void    SetDozenNum(int a_dozen_num);
        int     GetDozenNum();
        void    SetStatus(int a_status);
```

```
int     GetStatus();
void    SetStime();
int     Decomposition(int qty, double stime[]);
};
```

The next function is called *GetStyle*, which retrieves style data from the *STYLE* table. It opens the *STYLE* table, searches for the style which has a matched style ID, and returns all data of the style including the operation structure and their ID. Its data structure is shown as below:

```
class Style {
        private:
                int             style_id;
                OpStyle         *root;
                OpStyle         *op_style_array;
                int             op_style_num;

        public:
                Style();
                Style(int a_style_id);
                ~Style();
                void    SetId(int a_style_id);
                int     GetId();
                void    SetRoot(OpStyle*);
                void    GetRoot(OpStyle*&);
                void    SetNumOpStyle(int);
                int     GetNumOpStyle();
                void    SetOpStyleArray(OpStyle*);
                OpStyle* GetOpStyleArray();
                void    MidTravel(OpStyle* a_opstyle, OpStyle opstyle[], int& n);
                void    AllOptype(int typelist[], int& n);
        };

class OpStyle {
        private:
                int             op_id;
                OpStyle    *prev1, *prev2, *prev3;
                OpStyle         *next;
                float   stime;

        public:
                OpStyle();
                OpStyle(int a_op_id);
                ~OpStyle();
```

```
        void    SetId(int a_op_id);
        inn     GetId();
        void    SetStime(double);
        double          GetStime();
        void    SetPrev(OpStyle*   a_prev1,   OpStyle*   a_prev2,
                OpStyle*a_prev3);
        void    GetPrev(OpStyle*&   a_prev1,   OpStyle*&   a_prev2,
                OpStyle*& a_prev3);
        void    SetNext(OpStyle* a_next);
        void GetNext(OpStyle*& a_next);
};
```

The eighth function is called *IsOverTime*, which evaluates worker's overtime status, which may be -1, 0, +1, indicating that the worker is unavailable, available but un-committed, or committed. The next function is called *Mps*, which creates a master production schedule under a given capacity level. The capacity is set at three levels: 0,1,and 2 for the current planning week, indicating that the plant capacity is normal, uses overtime and subcontract, or include new hire. This function updates the following five tables: *UMPS, UMPS_WK, MISCELL, WK_OPTP*, and *JOB*. The *RecoverFiniQty* function allocates jobs and restores the value assigned to the initial finished quantity during the iterative planning process. The *Select* function picks the most urgent operation type to support the Umps function, which assigns workers to each operation type. The most efficient worker is picked by the *SelectWorker* function.

The *UpdateCurrTrain* function updates current training days in the *WK_OPTP* table on a weekly basis. The function *UpdateFiniQty* is used to update finished quantity for each job in the JOB table. The *UpdateThisTrain* is a sub-function of *Umps* implemented to update the current training status in the *WK_OPTP* table. If one worker is assigned to an operation type, the training status of this worker is set to 1, otherwise set to 0. The *UpdateUmpsWk* function is a sub-function of Umps and is implemented to update the master production schedule in the *UMPS_WK* table. It opens the *UMPS_WK* table, sets worker ID and operation type ID with the information store in array *worker* as defined previous in class *worker*. The *UpdateUmpsDB* function is a sub-function of *Umps*. It is implemented to update job id and throughput in the *UMPS* table. It opens the *UMPS* table if the current planning week exists in the table. It deletes all the record, and then appends relative records such as week ID, job ID, throughput quantity of each job.

3.6 Detailed Production Scheduling Module

The detailed production scheduling module is implemented according to the scheduling procedure presented in chapter two. It generates a production plan which assign each worker to a station and a job (operation) based on the master production plan. It is programmed to create a daily plan for the immediate week only because the master production schedule is updated weekly and during each week many changes may render a static production less realistic. Similar to the MPS implementation, the input data for DPS are drawn from many tables. In addition to

those used by the MPS module, the master production schedule is retrieved from the *UMPS_WK* table and used as the basis for the detailed scheduling. As its output, the DPS module resets initial efficiency and current efficiency for each worker at each operation type in the worker to operation type (*WK_OPTP*) table. It generates a new *DPS_J_WK* table which stores worker's assignment to a workstation for a specific job and the daily throughput. It also updates work in process inventory for each job at each station in the *DPS_WIP* table.

The DPS module consists of twelve major modules. The first function is called *GetOperId*, which finds all operations for each operation type. The *IniJobTask* function initializes the task of jobs to operate the module. The function *IniUpperBound* initializes the inventory limitation. The *InputData* function is used to input data from the database and structure all the required data regarding the jobs, garments, operations, workers, and production schedules. The *JobIndex* and *OperationIndex* functions are used to find a job and a operation indices. The *ToDpsJWk* function outputs the detailed production schedule and creates the *DPS_J_WK* table. The output includes worker and job IDs, and operation type which the worker is assigned to.
The *WorkerIndex* function finds the worker's index when the worker's ID is known. The *WorkerRank* function sorts workers according to their efficiency based on each operation type. The *WorkerAssignJob* function assigns workers to each job. The rule is to assign the most efficient worker to the most urgent job. the urgency is updated after a worker is assigned. The *WorkerAssignOp* function assigns each worker to a workstation. It assigns the most efficient worker to the most urgent workstation. The workstation's urgency is updated after a worker has been assigned.

3.7 Plant Capacity Planning Module

The capacity planning module is implemented according to the procedure presented in the previous chapter. It retrieves the current week from the *MISCELL* table, checks the *JOB* table for subcontractors status, and gets the *WORKER* table for identify workers' overtime availability and planned absence. Decisions made by the capacity planning module includes the subcontracted quantity of each job to each subcontractor, overtime commitments, and number of new hire for each operation type.

3.8 Material Flow Simulation Module

The material flow simulator consists of twelve functions. The *ShowOperation* function is implemented to create a shop floor layout which reflects the physical location of all workstations. It retrieves the location of each workstation as stored in the database and places workstation symbol on the computer screen accordingly. Because of limited display space on the monitor, the idea of a physical layout display is modified to maximize the use of the monitor space. This function calculates the amount of space available for each station based on the total number of stations in the system. It also allocate space for display of raw materials and finished goods. The *ShowOneOperation* function is a sub-function of *ShowOperation*, which is developed to detail the display of one workstation at a time to complete the physical layout display. It

outlines each workstation, labels it, and displays its current manning and WIP inventory levels. A physical layout example is shown in Figure 18.

The *ShowTree* function is created to display a logical layout of a given production shop floor, in which workstations are arranged according to the operation sequence of a selected job. If more than one job is on the production shop floor, multiple logical layouts are generated and displayed. This function retrieves the operation tree for each job, evaluate operation relationship, estimate the size for each logical layout, and assign a relative location for each operation on the tree, create a frame for each station, and connect them. This function is different from the *ShowOperation* in that it ignores all workstations which are not required for the target job. The *JobTree* function is a sub-function of ShowTree. Upon request, it creates a display of a workstation as specified. The workstation display is similar to the ones in the physical layout except that it only show the information related to the job. A logical layout example is shown in Figure 19.

The *SimMenu* function is implemented to organize and display command functions available to the user in a menu format. The text contents of menu items are stored in two-dimension array of the menu. The *Simulation* function is the main function of this module which simulates the material flow of the production system and dynamically updates the graphic display of the manning and WIP inventory levels at each workstation. It works with both physical and logical layouts. During a simulation session, the worker assignment and inventory levels are initialized with the current data retrieved from the database and then updated dynamically with projected events and the user's input (i.e., worker re-location). To enable the user to control the system, the *WhichWorker* function is implemented, which identifies the worker whose icon on the display is picked by the user. Then the *OneWorker* function is activated to create a graphical display of the selected worker in the workstation newly designated by the user. The *ZoomOper* function allows the user to zoom in and out of a workstation to better view the station. An enlarged workstation (#ID14-3) layout is shown in Figure 19.

The *OneWorker* function is used to create a graphical display of a worker in a specified workstation. When requested, it calculates the location of the worker in the given station and creates a worker symbol. It paints the worker in a color selected to represent his/her current efficiency level at the workstation. When the worker is relocated to a new station, this function re-evaluates its efficiency at the new station and change the color accordingly.

The *TextOper* function is used to display additional data about a selected workstation in a text form. As shown in Figure 20, it presents a workstation ID, its assigned operation type, the number of jobs assigned to the workstation, the inventory level of each job at the station, the throughput of each job, the number of workers assigned to the station, their ID and skill level at the station.

The *WeekTask* function is used to summarizes the quantity of each job scheduled for this week. It displays the quantity of each job to be finished in the current week, as obtained from the master production schedule, in a two-dimensional graph, where the X axis represents jobs and

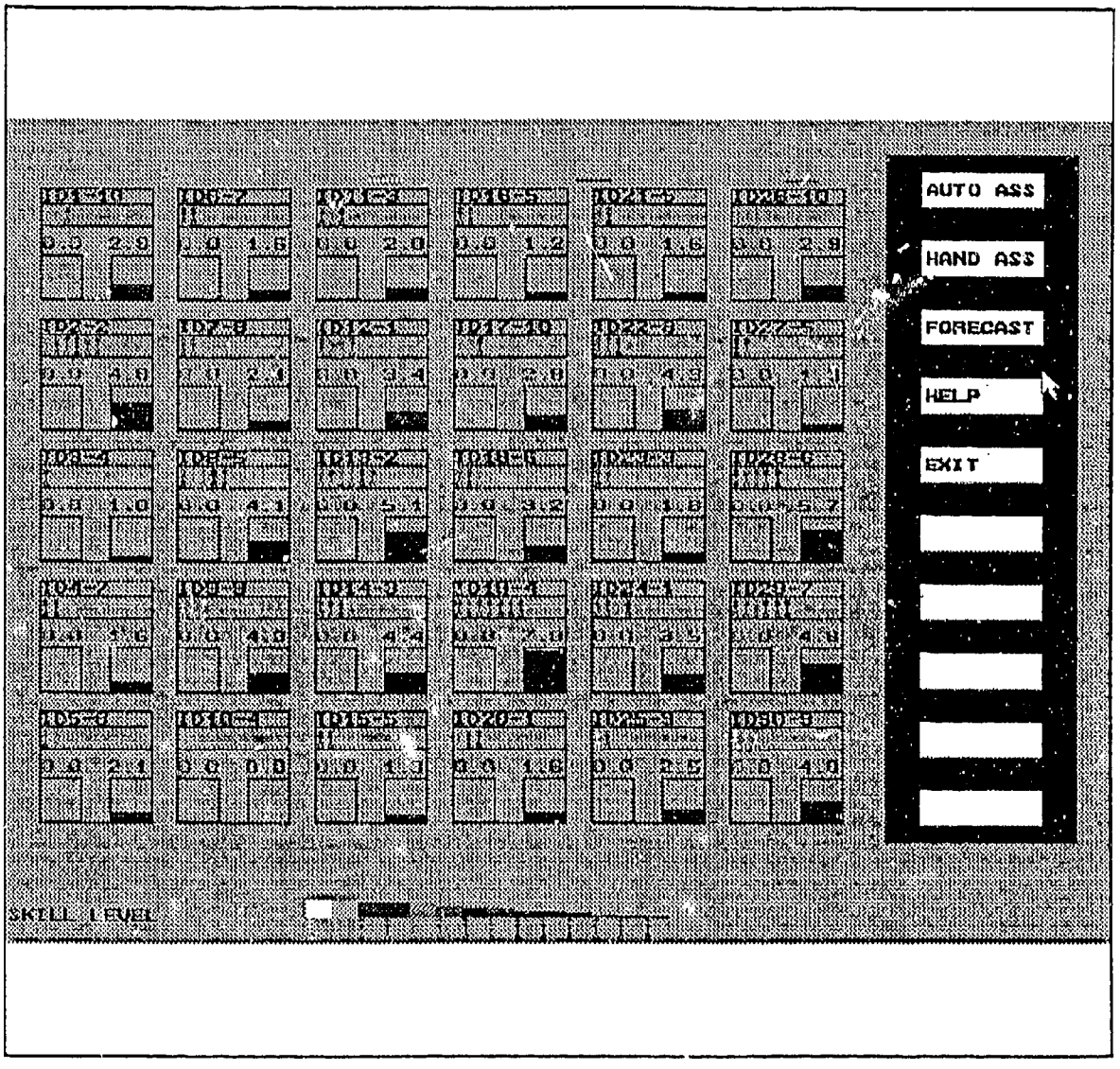


Figure 18 : A Physical Layout

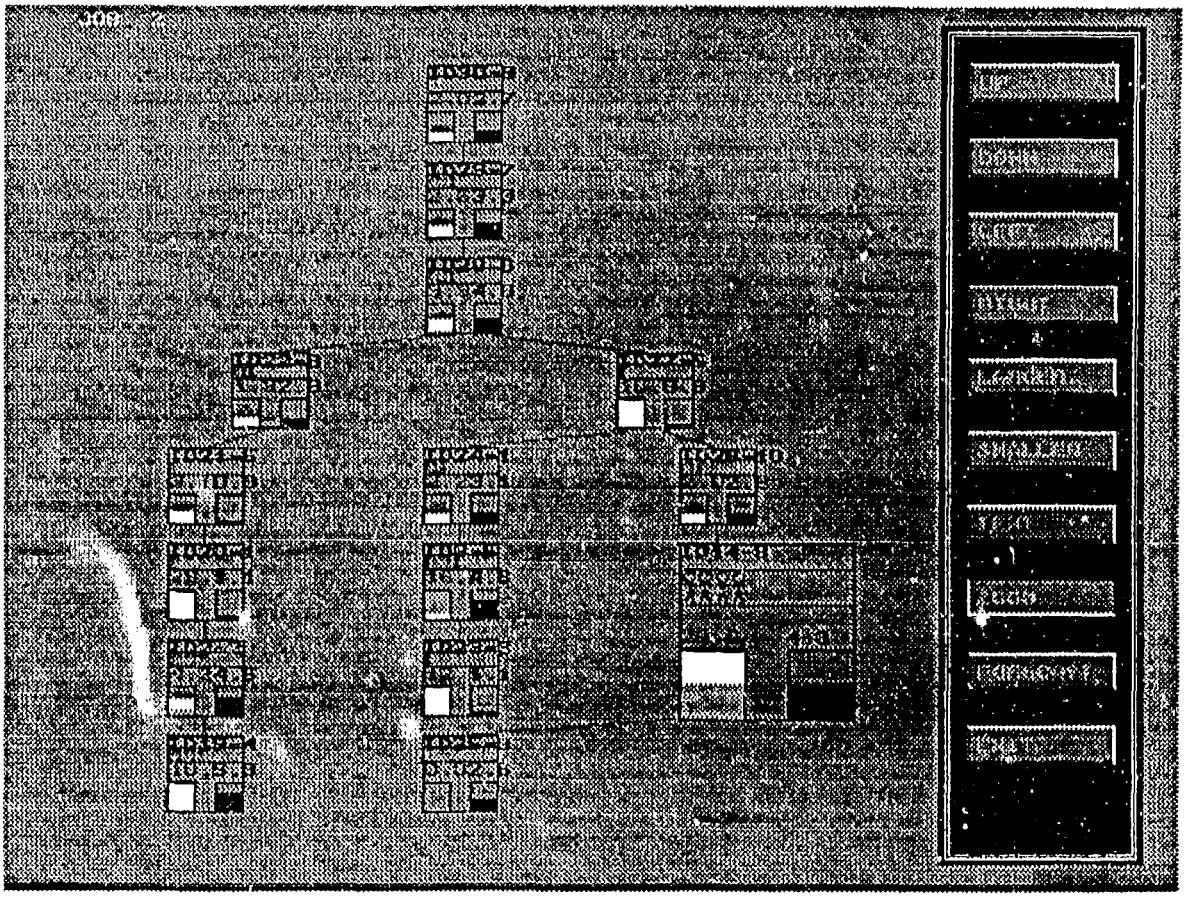


Figure 19 : A Logical Layout

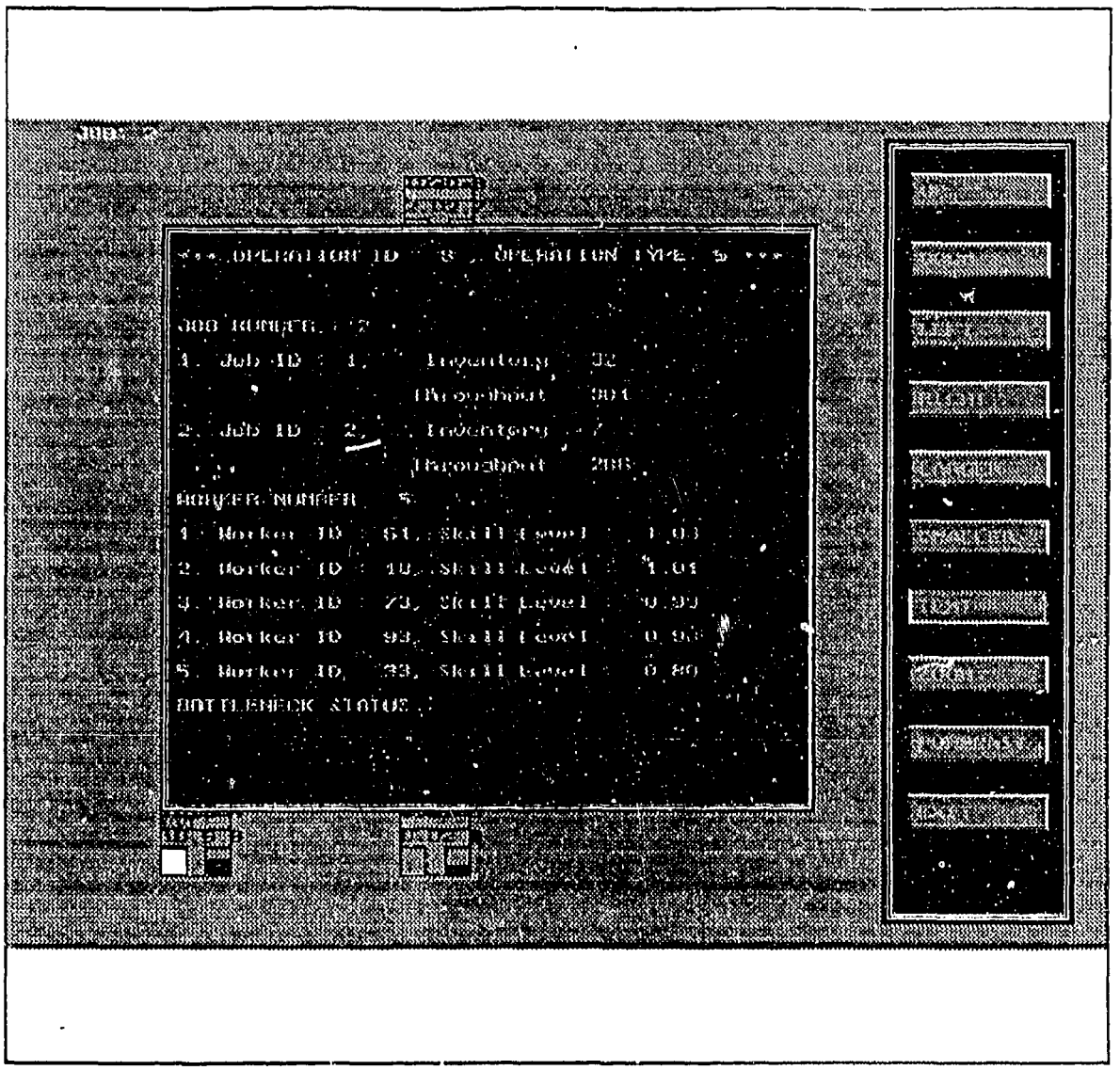


Figure 20 : Text Information for a Workstation

the Y axis does the number of bundles of each job. The *WorkerPost* function shows the worker assignment in the current week. It displays the assignment of each worker by operation type, as obtained from master production schedule.

3.9 System Application and Validation

At the design and development stage, many apparel production managers at different companies helped in contributing production and operation data and in sharing ideas for building a friendly and practical planning system. At the implementation stage, the system prototype was periodically showed to the managers at Bernard Cap, who evaluated the system and provided feedback for changes and improvements. This system was also applied to a number of application instances in a bundle production environment. In one of the examples, four jobs in two orders are processed with 100 workers at 30 workstations. To run the system, the database was first loaded with required production and resource data, as listed in Appendix 3. The operation sequences of the jobs are captured in Figure 21. Examples of system outputs are given in Figures 22-25. A summary of the 16-week master production schedule is shown in Figure 22. A 3D snapshot of the master production schedule for week 5 is shown in Figure 23. The detailed production plan for the first work day in the current week is shown in Figure 24. Note that the throughput do not reflect the utilization level of each workstation. Figure 25 shows the inventory level at each workstation at the beginning of the first day.

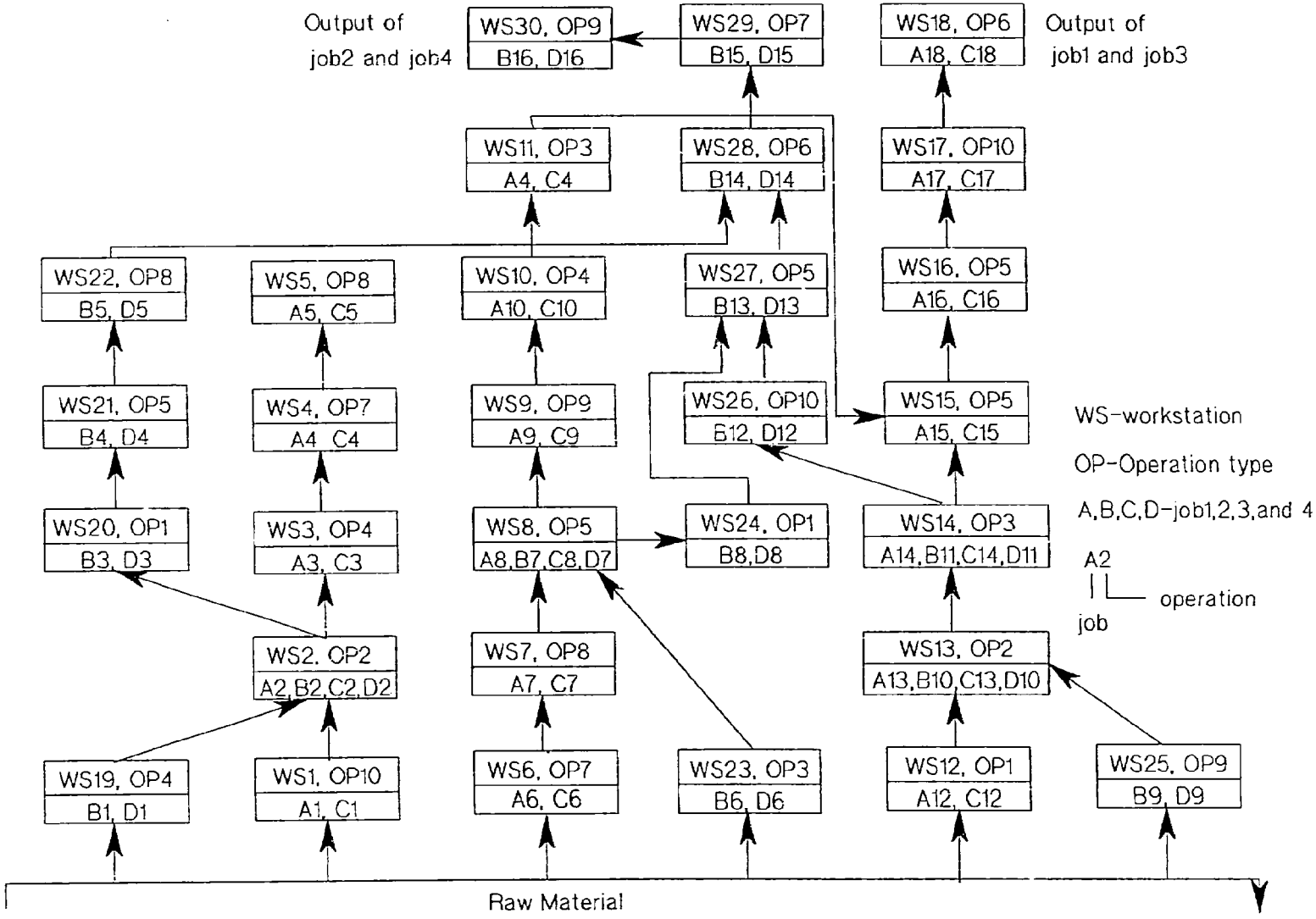


Figure 21: Operation Sequence of Jobs

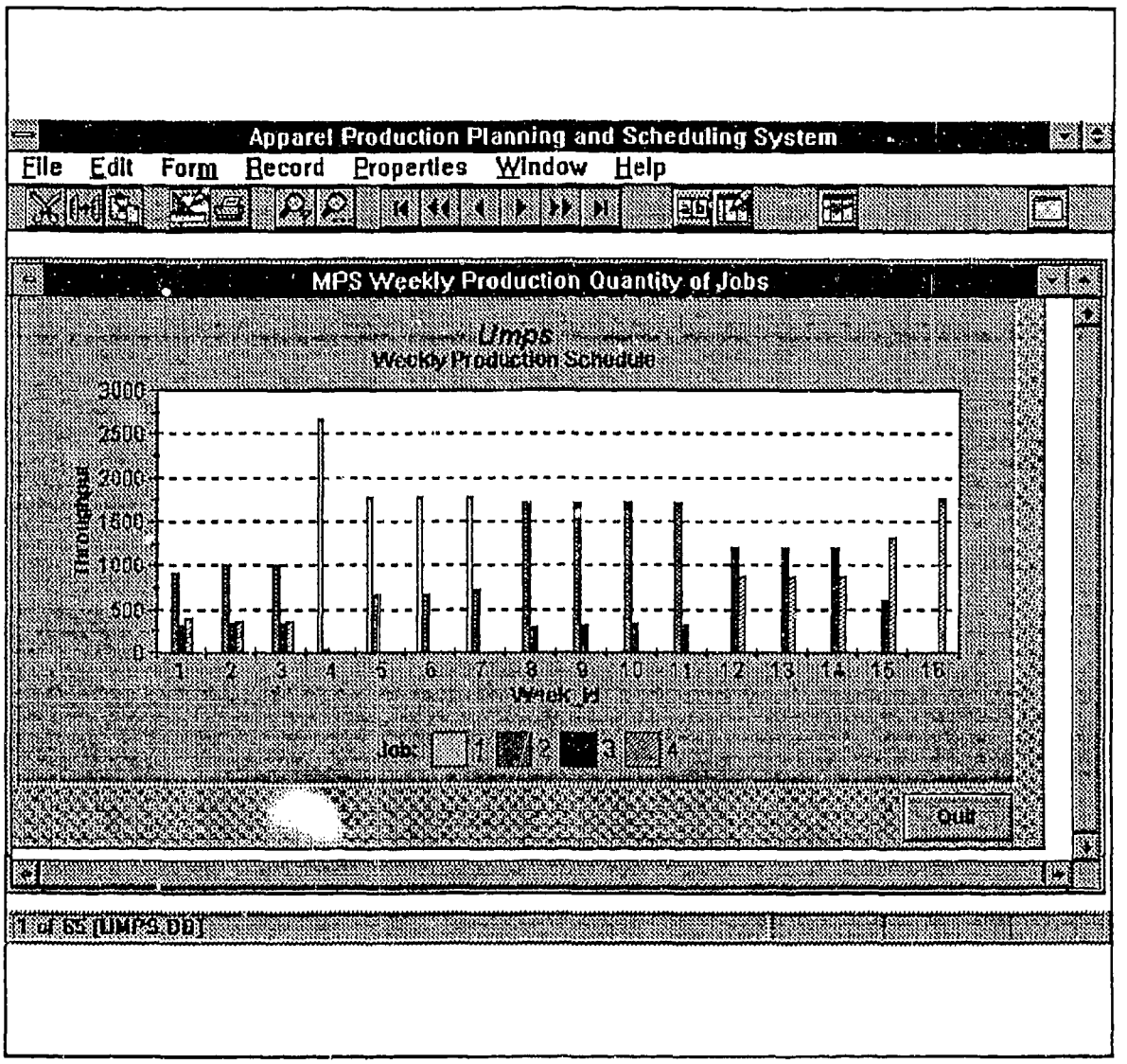


Figure 22 : A Master Production Schedule

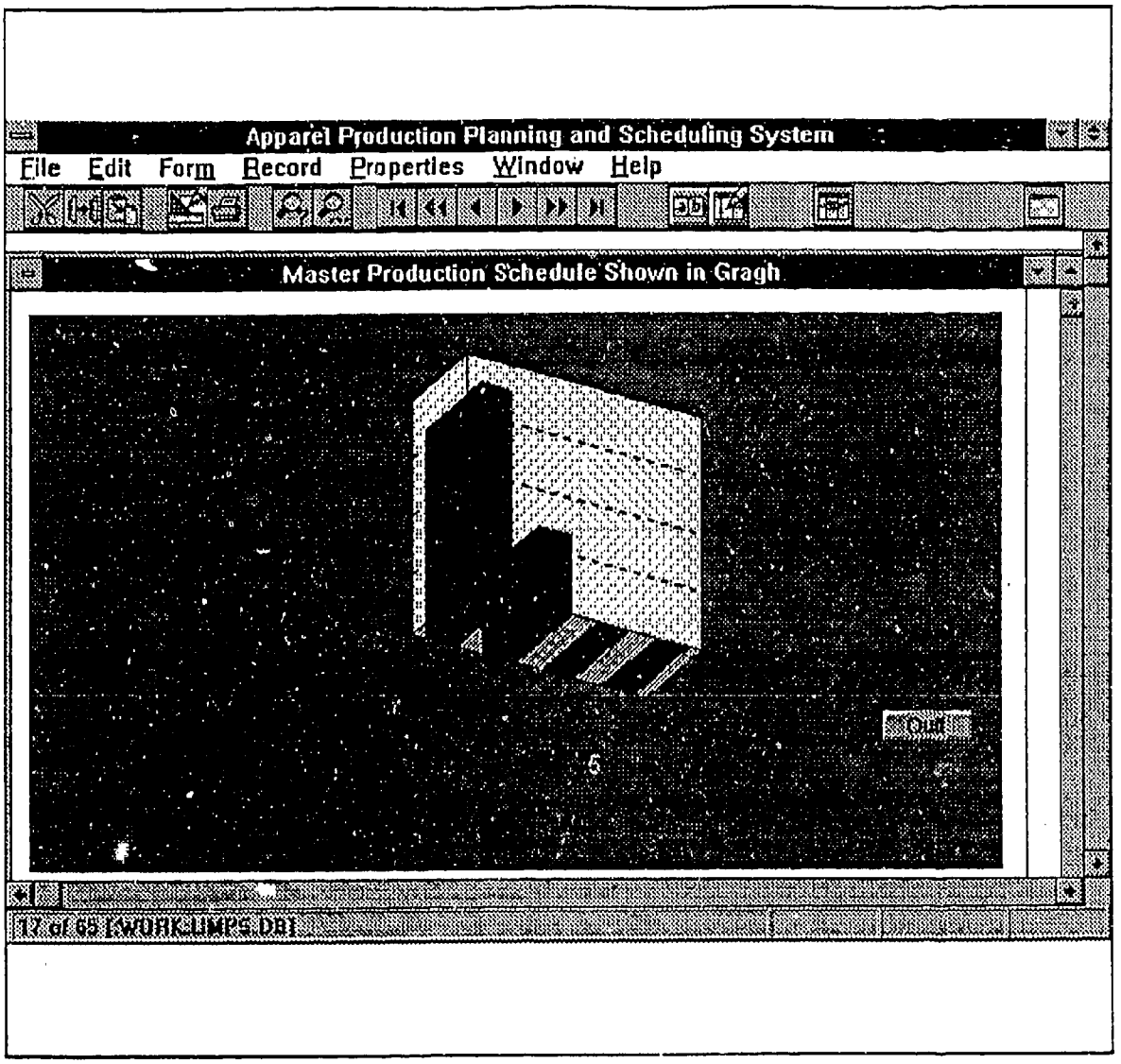


Figure 23 : An MPS Snapshot

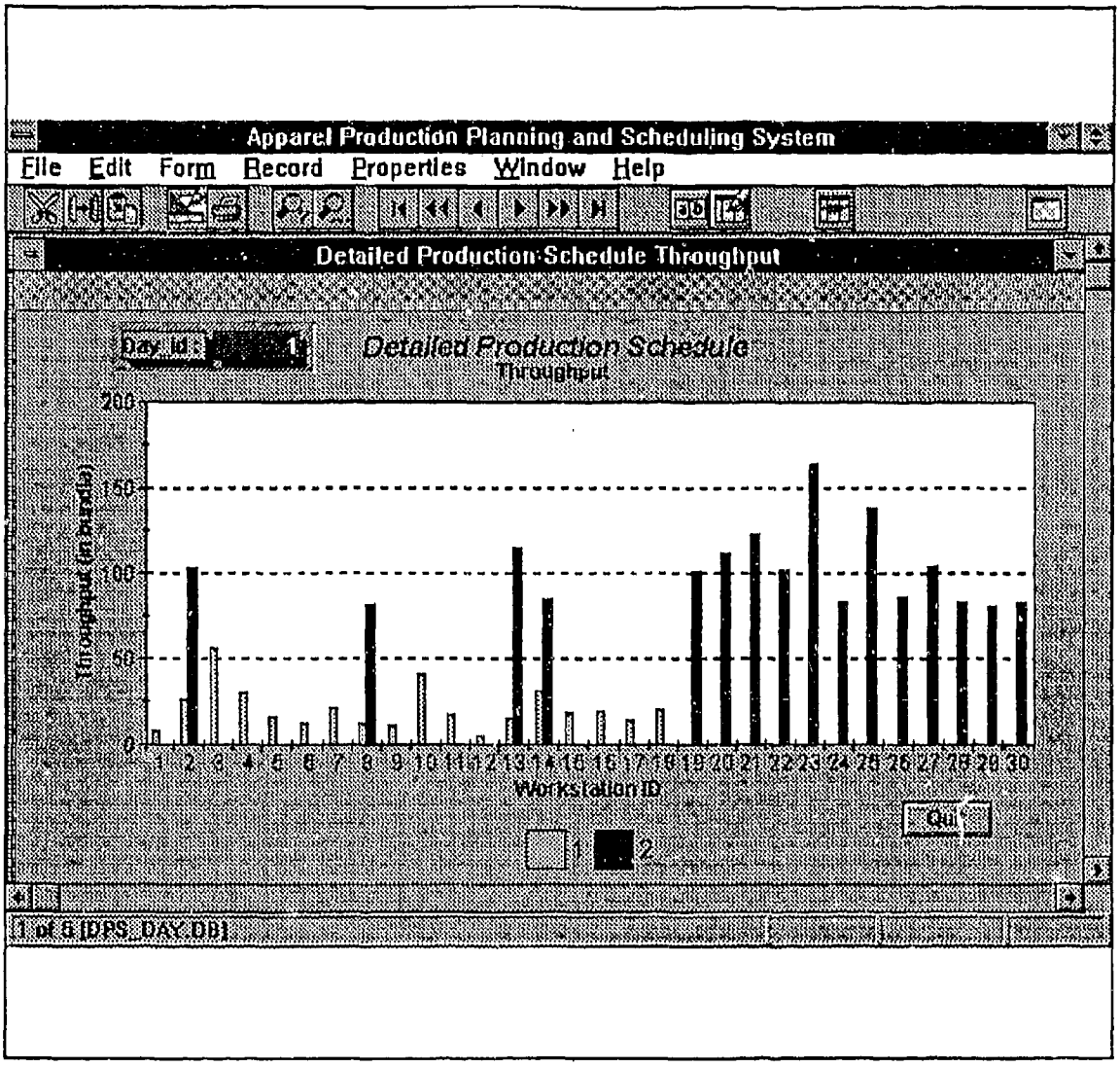


Figure 24 : Detailed Production Plan in the First Day

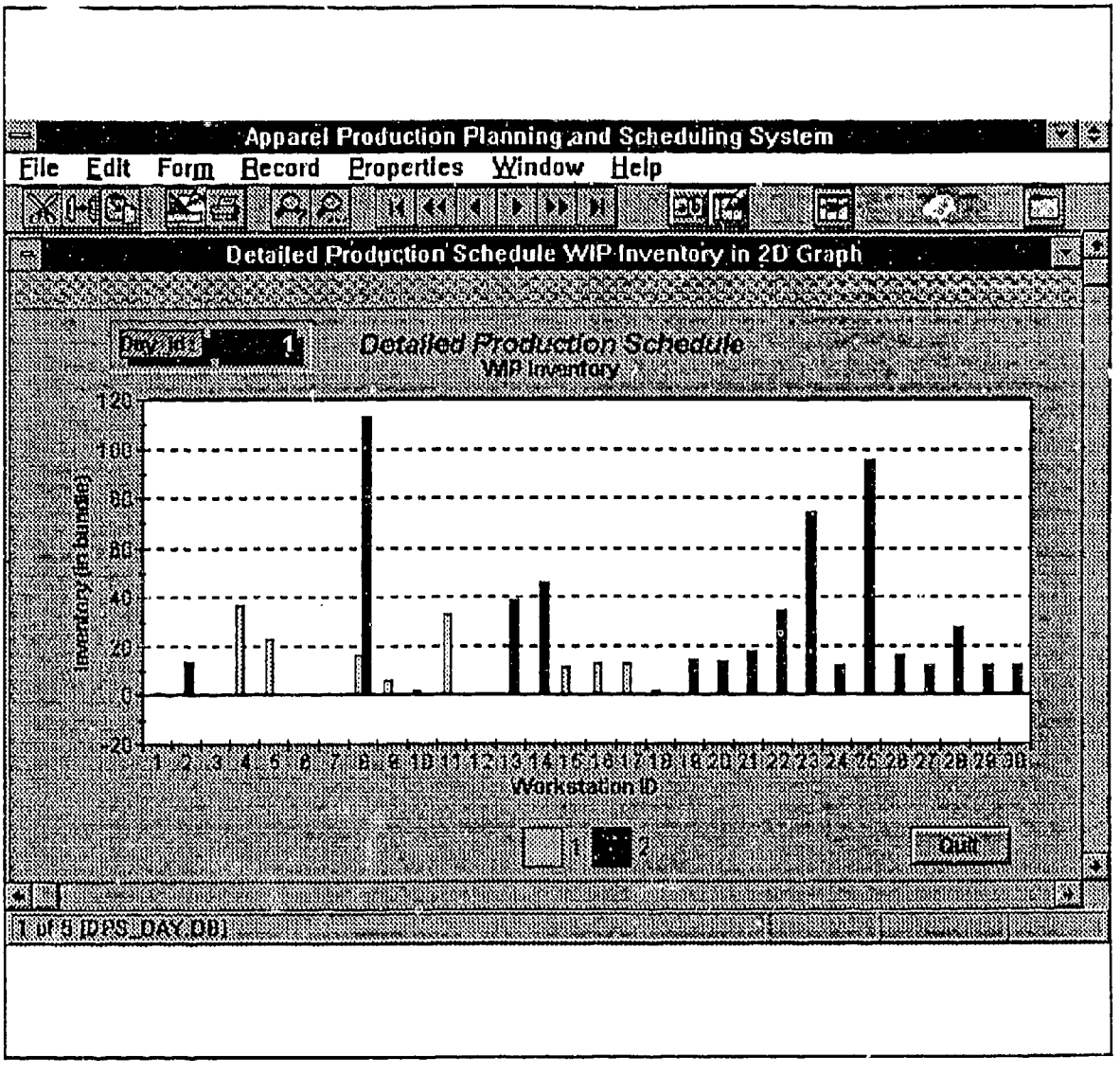


Figure 25 : Inventory Level on the First Day in Current Week

4. CONCLUDING REMARKS

This chapter concludes the research project in two sections. The first section summarizes the results. The second section elaborates potential projects for further research.

4.1 Summary

The objective of this research project is the development and implementation of a practical production planning and scheduling system for the apparel manufacturing industry as a quick and accurate decision tool. In this study, the apparel problem is identified a make-to-order production problem. More specifically, an assemble-to-order problem with no inventory (both finished and semi-finished goods) to smooth production. Due to the nature of short production runs, use of general purpose machines, multiple and progressive sewing skills, the existing scheduling and line-balancing techniques can not be applied to solve the unique scheduling problem. Because of the complexity and dynamics of the production system, any momentarily optimal solution is not important and inherently difficult to obtain.

In this study, we takes a hierarchical approach to the problem and to the development of the integrated production planning and scheduling system. A commercial database management system is used to ensure the integrity of apparel product and production data. In addition to the common database and a graphic user interface, the system consists of six major application modules. They include: a master production planner, a detailed production scheduler, a plant capacity planner, a production method evaluator, and a material flow simulator. Through the graphic user interface, the user manages the database and operates the application modules to make production decisions. The master production scheduler creates a weekly production plan for all jobs specified in each order. The detailed production planning module decomposes a master production plan into worker assignment and job routing on a daily basis. The plant capacity planning module evaluates overtime, subcontract, and new hire options for additional resources to ensure meeting delivery commitments. The material flow simulator further validates the static production plans under a given production environment. The production method evaluator enables the user to assess the appropriateness of the adopted bundle production method.

With the proposed system, manufacturing lead time can be quickly evaluated, at accepting a new order. Realistic production plans can be generated quickly and consistently and be relied upon for promoting long-term growth of worker's sewing skills and plant capacity. The material flow simulator not only validates static production plans but also allows the user to evaluate the impact of worker re-assignment in the process of improving material flow. The simulator also summarizes current worker assignment and WIP levels in each station at any time.

The system can uniquely project worker's skills at any operation type based on a pre-determined worker assignment plan, and estimate the time-phased plant capacity accordingly. This feature allows the user generate accurate production plans while being able to improve workers' long-term productivity. The skill prediction model was developed based on real apparel production and operation data collected from the industry over a two year period during the

project. Regarding the planning and scheduling procedures implemented in this system, although mathematical models and analytical solution procedures were attempted in this project, the final solution procedures are primarily based on shop loading principles such as job urgency and load balance. These algorithms are basically heuristic in nature and quite efficient. The master production scheduler and the plant capacity planner both schedule production by decomposing production requirements to operation type level to improve accuracy, instead of adopting a typically used rough cutting approach. The planning and scheduling system is also more efficient than any simulation based scheduling approach. To improve production plans, discrete event simulation techniques are used to evaluate the production plans.

The design of the relational database is a unique feature of this system. It captures apparel, production, and resource data. has the knowledge of relating orders to jobs, garment styles, operations, operation types, workstations, workers, production schedules, and WIP inventory levels. It has the structure for further expansion and can be easily used to support other system applications such as MRP and payroll. In addition to production plan validation, the material flow simulator could also be used as a production control tool, when linked to an on-line shop floor control system. Upon completion, the user could monitor WIP inventory built-ups and re-locate workers to alleviate the problem. The simulator configure a plant layout for each simulation session, using job and resource information retrieved from the database including the location of each workstation. Therefore the simulator can model any plant configuration without input from the user.

4.2 Future Research Work

The system can be further improved and enhanced. A separate sewing skill prediction model, for example, could be developed to accurately predict the skill of a sewing worker who is being switched from one operation to another. The skill prediction model currently implemented in this system is for workers without prior sewing experience. It could be further enhanced by introducing operator related factors to the model. A larger data set could also red· · the unexplained variations in the current skill prediction model. The proposed system is focused on the planning and scheduling problem in a bundle production system. It could be extended to a multiple production facility environment, in which different production methods such as modular and unit production are used. The utility of this system should also be evaluated for the dual-use production environment where relatively different products such as a mix of handbags and garrison caps could be manufactured at the same time.

Additional system applications such as manufacturability evaluation of a product design, shop floor control, payroll, accounting, and MRP could be implemented and integrated into the system and share the same database, leading to the development of a concurrent engineering environment, in which product and process design activities are closely coupled. The system could be also greatly enhanced by incorporating production and scheduling expertise into the system to increase its intelligence and reduce its reliance on the user for, say, manual worker re-location in response to inventory build-ups on the shop floor.

REFERENCES

1. Adam, N.R., J. W. M. Bertrand, D.C. Morehead, and J. Surkis, "Due Date Assignment Procedures with Dynamically Updated Coefficients for Multi-Level Assembly Job Shops", European Journal of Operational Research, 68, 1993, pp.212-227.

2. Baker, K.R., "Sequencing Rules and Due-Date Assignment in a Job Shop," Management Science, Vol. 30, 1984, pp. 1093-1104.

3. Baker, K.R., "Introduction to Sequencing and Scheduling," John Wiley and Sons, New York, 1974.

4. Bedworth, D.D., and J.E. Bailey, Integrated Production Control Systems, Management, Analysis, Design, John Wiley & Sons, Inc., New York, 1982.

5. Cerny, V., "Thermodinamical Approach to the Travelling Salesman Problem: An Efficient Simulation Algorithm", Journal of Optimization Theory and Applications, Vol. 45, 1985, pp. 41-51.

6. Chen, C., Racine R., and Swift F., "A Practical Approach to the Apparel Production-Planning and Scheduling Problem," International Journal of Clothing Science and Technology, Vol. 4 1992, pp. 9-17.

7. Chen, C., Racine, R., and Swift, F., "A Study of The Production Planning and Scheduling Problem in the Apparel Manufacturing Industry, Phase I," final technical report, (DLA900-91-C-1481), Defense Logistics Agency/Defense Electronics Supply Center, October, 1991.

8. Cheng, T.C.E., and Gupta, M.C., "Survey of Scheduling Research Involving Due Date Determination Decisions," European Journal of Operations Research, Vol. 38, 1989, pp. 156-166.

9. Connoly, D.T.,"An Improved Annealing Scheme for the QAP", European Journal of Operational Research, Vol. 46, 1990, pp. 93-100.

10. Doctor, S.R., T.M. Cavalier, and P.J. Egbelu, "Scheduling for Machining and Assembly in a Job Shop Environment," Operation Research, vol.31, 1993, pp.1275-1297.

11. Fry, T.D., Oliff M.D., E.D. Minor, and G.K. Leong, "The Effects of Product Structure and Sequencing Rule on Assembly Shop Performance," International Journal of Production Research, 1989, vol.27, no.4, pp.671-686.

12. Fumero, F., and C. Vercellis, "Capacity Analysis in Repetitive Assemble-to-Order

Manufacturing Systems," European Journal of Operational Research, 78)1994), pp.204-215.

13. Gessner, S.A., Master Production Schedule Planning, John Wiley & Sons, New York, 1986.

14. Goodwin, J.S., and J.K. Weeks, "Evaluating scheduling Policies in a Multi-Level Assembly System," International Journal of Production Research, 1986, vol.24, no.2, pp.247-257.

15. Goodwin, J.S., and J.C. Goodwin, "Operating Policies for Scheduling Assembled Products," Decision Science, vol.13, 1982, pp. 585-603.

16. Graves, S.C., "A Review of Production Scheduling", Operations Research, vol.29, no.4, July-August, 1981, 646-675.

17. Harrison, T.P., and J.E. Ketz, "Modeling Learning Effects via Successive Linear Programming", European Journal of Operational Research, Vol.40, 1989, pp.78-84.

18. Hendry, L.C. and Kingsman, B.G. "Production planning systems and their application to make-to-order companies", European Journal of Operational Research, 1989,40, 1-15.

19. Hiller, R.S., and S. Shapiro, "Optimal Capacity Planning When There Are Learning Effects", Management Science, 32, 1986, pp.1153-1163.

20. Huang, M.D., Romeo, F., and Vicentelli, A.L., "An efficient General Cooling Schedule for Simulated Annealing", IEEE Int. Conference on Computer-Aided Design, Santa Clara, 1968, pp. 381-384.

21. Johnson, D.S., Aragon, C.R., McGeoch, L.A., and Schoven, "Optimization by Simulated Annealing: An Experimental Evaluation Part I, Graph Partitioning", Operations Research, Vol. 37, No. 6, 1989, pp. 865-892.

22. Johnson, D.S., Aragon, C.R., McGeoch, L.A. and Schoven, "Optimization by Simulated Annealing: An Experimental Evaluation Part II, Graph Coloring and Number Partitioning", Operation Research, Vol.39, No. 3, 1991, pp.378-406.

23. Ketcham, M.G., R.E. Shannon, and G.L. Hogg, "Information Structures for Simulation Modeling of Manufacturing Systems," Simulation, Feb. 1989, pp.59-67.

24. Laarhoven, P. J. M., Aarts, E. H. L. and Lenstra J. K. 1992, "Job Shop Scheduling by Simulated Annealing", Operations Research, Vol. 40, p.113-125.

25. Lewis, [illegible], J.R. Sweigart, and R.E. Markland,"Master Scheduling in Assemble-to-Order

Environments: A Capacitated Multiobjective Lot-Sizing Model", Decision Science, vol.23, 1992, pp.21-43.

26. Meredith J.R., The Management of Operations, John Wiley & Sons, New York, 1987.

27. Morton, T.E., and D.W. Pentico, Heuristic Scheduling Systems, John Wiley & Sons, Inc., New York, 1993.

28. Panwalkar, S.S., and Iskander, W., "A Survey of Scheduling Rules," Operations Research, Vol. 25, No. 1, 1977, pp. 45-60.

29. Philipoom, P.R., R.S. Russell, and T.D. Fry, "A Preliminary Investigation of Multi-Attribute Based Sequencing Rules for Assembly Shops," International Journal of Production Research, vol.29, no.4, pp.739-753.

30. Ploszajski, G., M.G. Singh, and K.S., Hindi, "An Overview of some Computer-Aided Production Management Issues", Information and Decision Technologies, 18 (1993), pp.405-413.

31. Pratsini, E., J. Camm, and A.S. Rxaturi, "Effect of Process Learning on Manufacturing Scheduling", Journal of Computer & Operation Research, Vol.20(1), 1993, pp.15-24.

32. Racine, R., C. Chen, and F. Swift, "The Impact of Operator Efficiency on Apparel Production Planning", International Journal of Clothing Science and Technology, Vol.4(2/3), 1992, pp.18-25.

33. Reeves G.R. and J.R. Sweigart,"Product-mix Model When Learning Effects are Present", Management Science, Vol.27, 1981, pp.204-212.

34. Rolling, Z., "A Study of the Annealing Schedule for the Job Shop Problem," a master thesis, Florida International University, 1994.

35. Russell, R.S., and B.W. Taylor III, "An Evaluation of Sequencing Rules for an Assembly Shop," Decision Science, vol.6, 1985, pp. 196-212.

36. Smith, J., Learning Curve for Cost Control, Norcross, GA: Institute of Industrial Engineers, 1989.

37. Townsend, M.A., and T.W. Lamb, "Detailed Simulation of A Real World Job Shop with Subassembly Requirements," Simulation, Aug. 1991, pp.114-128.

38. Vasta J.A., Understanding Data Base Management Systems, Wadsworth Publishing Company, Belmont, California, 1985.

39. Walpole, R.E., and R.H. Muers, Probability and Statistics for Engineers and Scientists, New York: Macmillan Publishing Company, 1989.

40. Wong, D.F. and Liu, C.L., "Floorplan Design of VLSI Circuits", Algorithmica, Vol. 4, 1989, pp. 101-107.

APPENDIX A1: USER'S MANUAL

USER'S MANUAL

FOR THE APPAREL PRODUCTION PLANNING AND SCHEDULING SYSTEM
developed as part of the project entitled

A STUDY OF THE PRODUCTION PLANNING AND SCHEDULING PROBLEM IN THE APPAREL MANUFACTURING INDUSTRY
PHASE II

(#DLA900-91-C-1481)

submitted to

Defense Logistics Agency
Manufacturing Engineering Branch
Cameron Station-DLA-PRM
Alexandria, Virginia 22304-6100
Attn: Mr. Dan Gearing

by

Dr. Chin-Sheng Chen
Industrial & Systems Engineering Department
Florida International University
University Park, ECS 416
Miami, Florida 33199
Tel: (305) 348-3753

March 31, 1995

TABLE OF CONTENTS

1. GENERAL DESCRIPTION

This manual provides a guide of usage of the apparel production planning and scheduling (APPS) system developed at Florida International University (FIU). The user is assumed to have basic scheduling knowledge such as master production scheduling, detailed production scheduling, and shop floor control.

The manual is organized in three chapters. An introduction of the system's functionality and architecture is given in Chapter 2. With the introduction, the user is expected to understand the functionalities supported by and implemented in the system. The third chapter details the operation instructions, which are written according to the system menus and supported by the system's graphic user interface (GUI). With this guide, the user should be able to run the APPS system in 30 minutes, provided that it is properly loaded on a PC platform.

2. SYSTEM SUMMARY

This chapter summarizes the system in three sections. The first section summarizes the functions the system is designed to perform. The second section summarizes how the system works and interactions among the system components. The third section specifies the system configuration.

2.1 System Functions

This system is developed for planning and scheduling production for an apparel bundle manufacturing system. It is aimed at better meeting delivery commitments and promoting long-term growth of both plant capacity and individual worker's sewing skills. This system has the capability of performing the following functions:

1) manage production and manufacturing information.
2) evaluate the feasibility of meeting a delivery date.
3) evaluate and select the best production method.
4) generate and maintain the master production schedule.
5) generate and maintain a detailed production schedule
6) evaluate static production plans.

2.2 System Operations

As shown in Figure 1, the system is composed of eight major modules: a database manager (DM), a sewing skill predictor (SSP), a plant capacity planner (PCP), a master production scheduler (MPS), a detailed production scheduler (DPS), a material flow simulator (MFS), a production method evaluator (PME), and a graphical user interface (GUI).

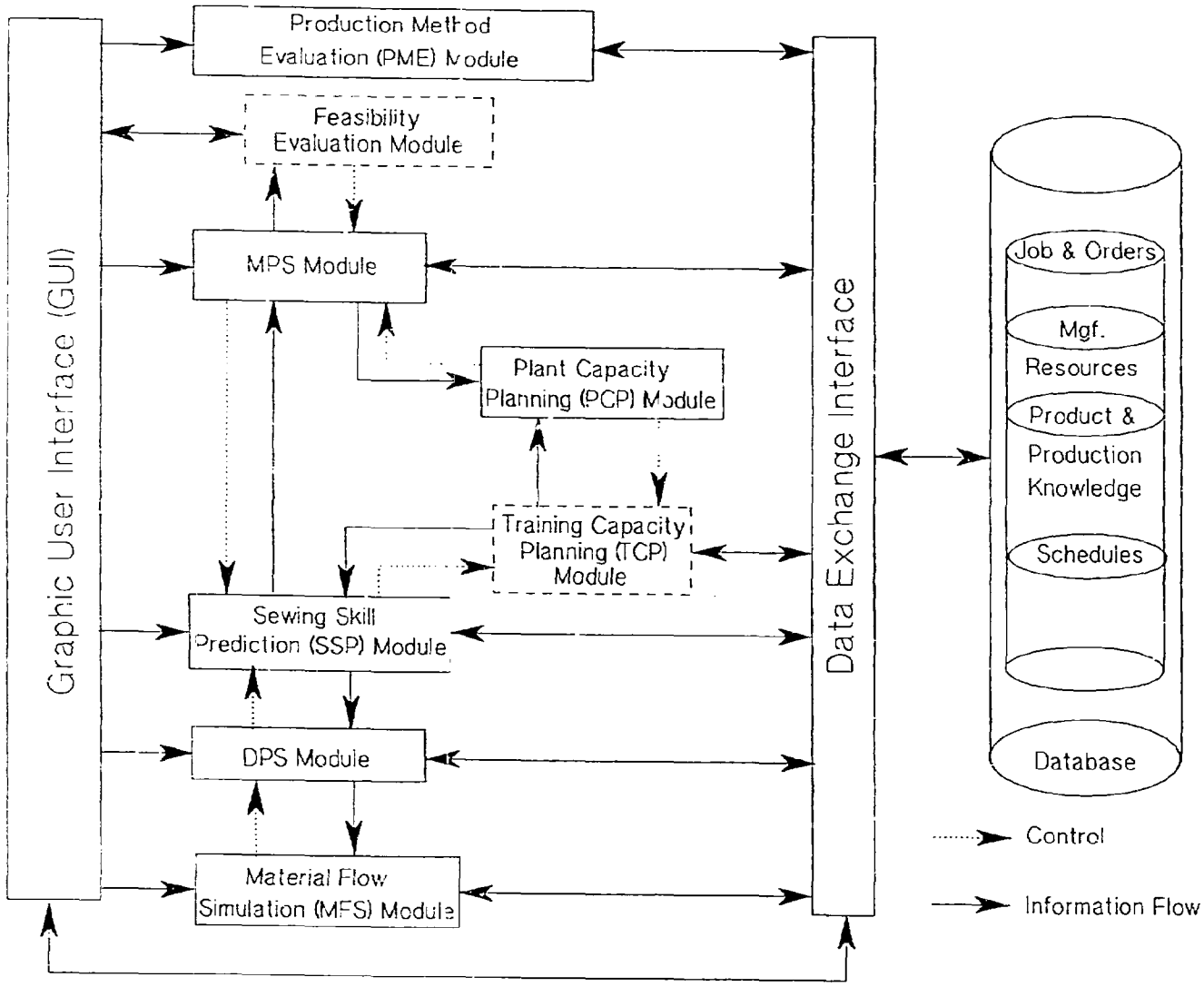


Figure 1: Proposed Apparel Production Planning & Scheduling System Architecture

The database manager is implemented to manage (add, delete and modify) apparel design, production, and manufacturing information stored in the system with a set of database management functions including browsing, searching, adding, deleting, modifying, and printing. The master production scheduling module retrieve job orders from the database and routinely generate/update the master production schedule, while ensuring sufficient manufacturing resources to meet delivery dates for all jobs and orders. The detailed production scheduling module takes the master production schedule, routes the jobs through workstations, and assigns sufficient workers to each workstation. The WIP inventory at each workstation is estimated for each job on daily basis. During the planning process, the sewing skill prediction model is called upon to assess the sewing efficiency of each worker at his/her assigned operation type and then estimate the time-phased plant capacity.

The skill predicti module predicts a worker's efficiency on any given day in training, with a set of sewing operation related coefficients previously stored in the database. The coefficients were extracted during the construction of the prediction model prior to implementation of this system. The material flow simulation module gives the user an effective tool to evaluate the validity of the static production plans in a given production environment. It provides a graphic display to show the dynamics of the simulation process and allows the user to interact with the system and re-assign workers to improve production flow. During each simulation session, the user can choose to view material flow in a physical layout or a logical layout. The GUI allows the user to manage all data through the database manager, invokes each system function, and audits intermediate and final results.

2.3. System Configuration

The system is implemented in Paradox in the DOS Windows environment. The Microsoft/DOS 3.1 is used to support the Paradox/Windows 4.5 Database Management System. The Paradox Engine 3.0 is used as the data exchange tool. The programming language used to implement the system is Turbo C++ Version 3.0. The computer platform is a Dell/386 PC. It requires 4MB RAM and a minimum of 30 MB hard disk. An EGA/VGA is required and mouse is a must.

The system requirements are: a PC/386 or better, 4MB RAM, 30 MB hard disk, an EGA or VGA monitor, a mouse, Paradox/DOS Windows 4.5 or higher, and Paradox Engine 3.0 or better. An AUTOEXEC.BAT file has been prepared to properly installed and set up required paths. No other specific configuration is required.

3. OPERATION INSTRUCTIONS

This chapter includes two sections. The first section outlines the procedure to initialize the apparel production planning and scheduling system. The second section details the guide for operating the system.

1 System Initialization

The system runs in Microsoft DOS Windows and Paradox Windows environment. It is recommended that the user first creates a directory in a hard drive for the program. Once in Paradox, the user can get to the directory and activates a file called MAINMENU.FSL to start up the system. After the software system is properly loaded to an appropriate computer system and supported by the Paradox, the user can start the system by just clicking a designated icon (currently marked as "FIU Apparel Schedule" at FIU). A standard window with a main menu on the top and a brief introduction will pop up. The layout of the entire menu is presented in Figure 2.

3.2 Operation Guide

This section is organized according to the proposed menu layout as shown in Figure 2. Each main menu is described in a subsection with all its sub-menu items. This section has six subsections highlighting commands under each of the following main menus: *Order, Resource, Knowledge, Schedule, SystemModules*, and *Exit*. The first four menus are designed for database management. The fifth menu provides an entrance to each application module; and the last one is designed for exiting the system.

3.2.1 *Order* Menu

This menu is selected by clicking the *Order* icon in the main menu. An *Order* menu layout will pop up and several its sub menus are then presented on a new window. These sub-menus are: *New, Edit, Delete, Print*, and *Quit*. Each is further explained below.

- *New*

  It adds a new order and its jobs to the system. Upon selecting this item, the system pops up a blank order form and ask the user to fill up. The new order's ID is automatically given by the system, but the following cells in the form need to be filled out by the user: *Customer_id, Order_date, Job_id, Style_id, Due_date, Penalty, Color_id, Size_id, Quantity, Finished_qty, Dozen_num, Status, Cost, Unit_price, Acceptability, Ini_finished_qty*, and *Committed_subcontract_qty*. One order may have multiple jobs. The user needs to fill out each job one at a time.

- *Edit*

  It is used to modify existing orders. Any cells except *Order_id*, and *Job_id* can be modified in the current order form.

- *Delete*

  It deletes an existing order and all its jobs. To prevent an order from being accidentally deleted, a dialogue box is provided which pops up after this command is clicked. It verifies the order ID with the user before deletion. It should be exercised with care, because it can not be recovered, after an order is deleted.

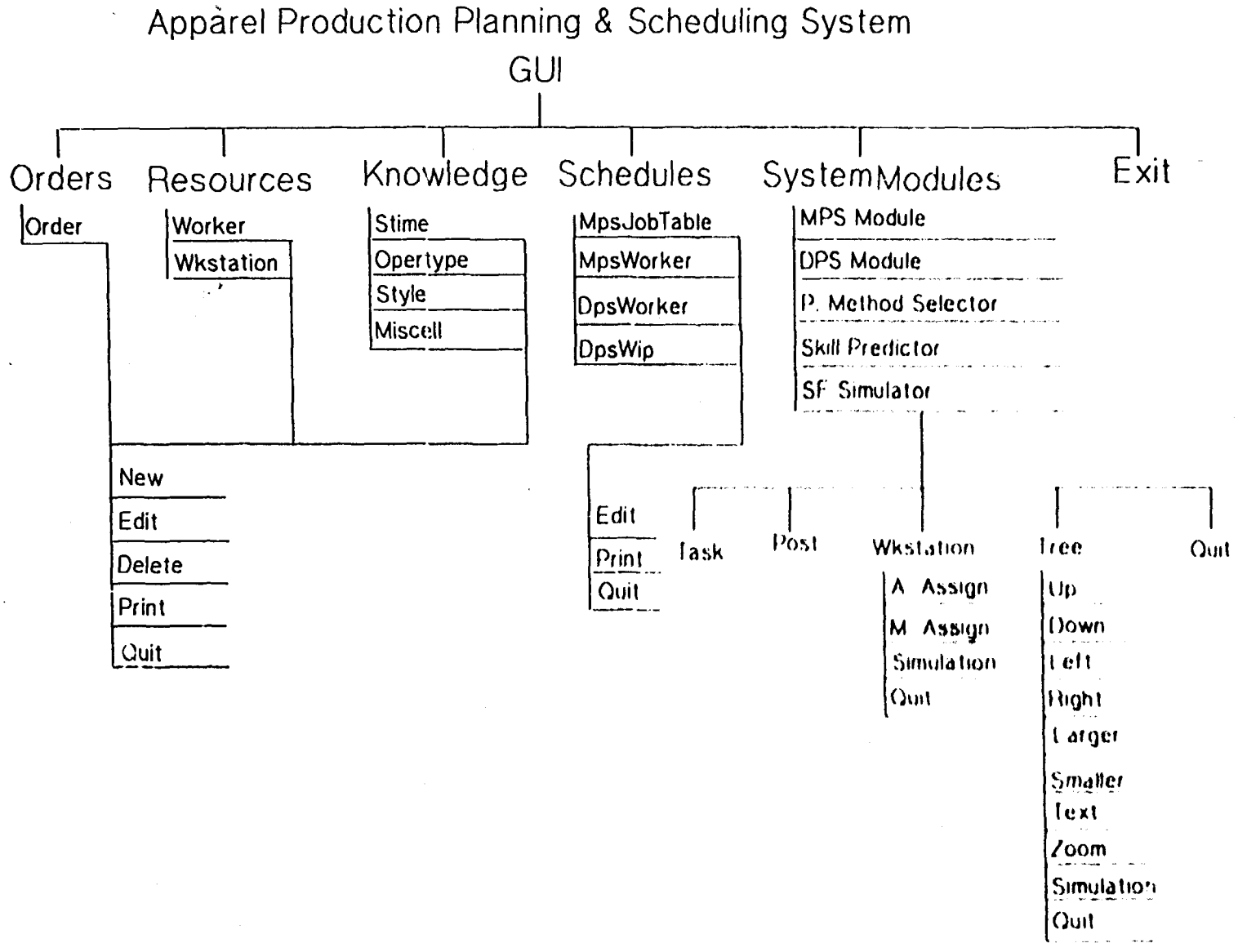


Figure 2 : System Menu Hierarchy

- *Print*

  It sends an order form to a printer. A report of an order is prepared by the system, and after triggered, the printer will print out the report.

- *Quit*

  It returns the control to the main menu.

3.2.2 *Resources* Menu

The *Resources* menu has two sub-menus. They are *Worker* and *Wkstation*.

- *Worker*

  By selecting this menu item, a window containing a form designed to display worker's information is shown on the screen. Five buttons are provided in the window for the user to add new workers and edit existing worker's data. These command items are:

  - *New*

    It allows the user to add new workers. After this item is selected, the system pops up a blank worker form and ask the user to fill out. The new worker's ID is automatically assigned by the system, but the remaining cells are filled out by the user: *Name*, *Hire_date*, *Optype_id*, *Current_training_days*, *Overtime1-10*, and *Absence1-5*. As a worker may perform several operation types, the form provides several columns to fill out the worker's skill level in terms of his/her training days at each operation type.

  - *Edit*

    It modifies the information about workers, one at a time. All cells except *Worker_id* in the worker form can be modified after this command item is clicked.

  - *Delete*

    It deletes an existing worker record. Before any deletion, a dialogue box will pop up after clicking this item to alert the user. Any deletion can not be recovered.

  - *Print*

    It generates a print file of worker information from the database and sends it to a designated printer.

  - *Quit*

    It returns the control to the main menu.

- *Wkstation*

  This command item is designed to add data for new workstations and edit existing

workstation records. Five control buttons are provided for the user to accomplish the task.

- *New*

  After this item is selected, the system pops up a blank workstation form and ask the user to fill out. The ID for the new workstation is automatically assigned by the system. The user need to only fill out the *Optype_id* cell in the workstation form.

- *Edit*

  It allows the user to modify existing workstation records. Since the workstation ID can not be edited, the only cell the user can modify is *Optype_id*. By specifying the workstation to be modified, a change to the operation type for the workstation is made.

- *Delete*

  It deletes an existing workstation record. After the command item is selected, a dialogue box pops up to confirm the deletion.

- *Print*

  It sends the data from a selected workstation form to a printer. A report for all the workstations is automatically prepared by the system and sent to a printer by the *Print* command.

- *Quit*

  It returns the control to the main menu.

3.2.3 *Knowledge* Menu

The *Knowledge* menu consists of four sub-menus. They are: *StandardTime*, *OperationType*, and *Miscellaneous*.

- *StandardTime*

  By selecting this item, the system pops up a form which includes the information about the standard processing time for each operation of each garment style and size. Five control icons are attached to the form for the user to add standard processing time for new operations and edit standard time for existing operations. The following commands are provided for the purpose.

  - *New*

    It adds new standard time. When this item is selected, the cursor points to a blank record and ask the user to fill out. The cells in the record to be filled out are: *Style_id*, *Size*, *Op_id*, and *Standard_time*.

- *Edit*

  It modifies existing records. When this control button is clicked, each cell in the form can be modified.

- *Delete*

  It deletes an existing record at a time according to the user's selection.

- *Print*

  When this control button is clicked, a report of standard processing times for all operations in the current form is automatically prepared and sent to a printer.

- *Quit*

  It returns the control back to its parent menu.

- *OperationType*

When *OperationType* is selected, a form designed for operation type is displayed on the screen. Five control buttons are provided on the form for the user to accomplish the following tasks:

- *New*

  It adds a new operation type to the system. When this button is selected, the cursor points to a blank record and prompts the user for a new operation type and its attributes. The ID of the new operation type is automatically given by the system, but remaining data are filled out by the system user. They are: *Name, F1-17, A, B, C,* and *Unit_cost.*

- *Edit*

  It allows the user to edit *OPTYPE* records. Any cells except *Optype_id* can be modified after this button is clicked.

- *Delete*

  It deletes an existing operation type. A dialogue box will pop up as this button is clicked to verify the deletion command with the user.

- *Print*

  It sends an operation type form to a printer. A report for all operation types is prepared automatically by the system and sent to a designated printer.

- *Quit*

  It returns the control to its parent menu.

- *Style*

When the *Style* sub-menu is selected, a form designed for *Style* is displayed on the

screen with five control buttons for the user to accomplish the following tasks:

- *New*

  It adds a new garment style to the system. When this button is selected, the cursor points to a blank form. It allows the user to key in data regarding all the operations for the new style. The ID of the new garment style is automatically given by the system, but the user needs to fill out the following cells: *Root_operation_id, Optype_id, Operation_id, Prev_op_id1, Prev_op_id2, prev_op_id3*, and *Next_op_id.*

- *Edit*

  It allows the user to modify existing data. Any cells except *Style_id* can be modified when this button is clicked.

- *Delete*

  It deletes an existing garment style record. After this button is clicked, a dialogue box pops up to make sure the right style to be removed from the system.

- *Print*

  It sends a style form to a printer. A report for the style is prepared automatically by the system and sent to a designated printer for printing.

- *Quit*

  It returns the control to its parent menu.

- *Miscellaneous*

  The Miscell sub-menu is created to update miscellaneous data. When activated, a form designed for miscellaneous information regarding the system is shown on the screen. Three control buttons are provided in the form for the user to conduct a number of tasks:

  - *Edit*

    It allows the user to modify each cell of the *MISCELL* table.

  - *Print*

    It enables the user to generate a report of the miscellaneous information.

  - *Quit*

    It returns the control to its parent menu.

3.2.4 *Schedule* Menu

The *Schedule* menu consists of a list of sub-menus. They are: *MpsJobTable, MpsJob2D, MpsJob3D, MpsWorker, DpsWorkerWkstationJob, DpsWipTable, DpsThroughput2D*, and

DpsWipInventory2D. The first four sub-menus are designed for handling the master production schedules, the other four sub-menus are designed for managing the detailed production schedules.

- *MpsJobTable*

  When the *MpsJobTable* sub-menu is selected, a form designed for master production schedule is shown on the screen. This form represents the master production schedule in the table. Three control buttons are attached to this form for the user to accomplish the following:

  - *Edit*

    It allows the user to edit an existing schedule. Each cell in the form can be modified.

  - *Print*

    It prepares a print file as shown on the screen and sends it a printer.

  - *Quit*

    It returns the control to its parent menu.

- *MpsJob2D*

  When the *MpsJob2D* sub-menu is selected, a form which represents the master production schedule is shown as a two dimensional bar chart. Only two control buttons are attached to this form:

  - *Print*

    It allows printing of the bar chart displayed on the screen.

  - *Quit*

    It returns the control to its parent menu.

- *MpsJob3D*

  When the *MpsJob3D* sub-menu is selected, a form which represents the master production schedule is displayed with a three dimensional graph shown on the screen. Only two control buttons are attached to this form:

  - *Print*

    It allows printing of the graph on the screen.

  - *Quit*

    It returns the control to its parent menu.

- *MpsWorker*

  When the *MpsWorker* sub-menu is selected, a form which represents the worker assignment for the master production schedule by week and operation type, is shown on

the screen. Three control buttons are attached to this form:

- *Edit*
  It allows the user to modify the master production schedule.

- *Print*
  It provides for the printing capability.

- *Quit*
  It returns the control to its parent menu.

- *DpsWorkerWkstationJob*

  When the *DpsWorkerwkstationJob* sub-menu is selected, a form, which represents the worker assignment in the detailed production schedule by operation type and workstation, is shown on the screen. Four control buttons are attached to this form:

  - *Edit*
    It allows the user to modify the assignment in the schedule.

  - *Delete*
    It allows the user to delete a record of the schedule or remove a worker from a schedule.

  - *Print*
    It allows the printing of the form on the screen.

  - *Quit*
    It returns the control to its parent menu.

- *DpsWipTable*

  When the *DpsWipTable* sub-menu is selected, a form, which represents WIP in the detailed production schedule by workstation and job in each week, is shown on the screen. Four control buttons are attached to this form:

  - *Edit*
    It allows the user to modify every cell in the data table.

  - *Delete*
    It allows the user to remove data from the table.

  - *Print*
    It allows printing of the table on the screen.

  - *Quit*

It returns the control to its parent menu.

- *DpsThroughput2D*

  When the *DpsThroughput2D* sub-menu is selected, a two dimensional bar chart which represents daily throughput as estimated in the detailed production schedule by job and workstation is shown. Two control buttons are provided:

  - *Print*

    It prints out the bar chart on the screen.

  - *Quit*

    It returns the control to its parent menu.

- *DpsWipInventory2D*

  When the *DpsWipInventory2D* sub-menu is activated, a two dimension bar chart representing the daily inventory level of each job at each workstation as estimated in the detailed production schedule for the current week is shown. Two control buttons are provided:

  - *Print*

    It prints the bar chart on the screen.

  - *Quit*

    It returns the control to its parent menu.

3.2.5 *SystemModules* Menu

The *SystemModules* menu provides an entrance to the application modules, which include: *MasterProductionScheduler*, *DetailedProductionScheduler*, *SewingSkillPredictor*, and *ShopFlowSimulator*

- *MasterProductionScheduler*

  It invokes the master production scheduling module. It can be activated only after all product, production, and manufacturing resources data are available.

- *DetailedProductionScheduler*

  It activates the detailed production scheduling module. This scheduler is usually activated after a master production schedule is prepared.

- *SewingSkillPredictor*

  It activates the skill prediction module which charts up the learning curve for each

operation type and predicts skill proficiency for estimating plant capacity during master production scheduling.

- *ShopFlowSimulator*

It invokes the simulation module to validate static production schedules. It has the following sub-menus: *Task*, *Assignment*, *Wstation*, *Tree*, and *Exit*.

- *Task*

  It displays the tasks scheduled in the MPS module for the current week.

- *Assignment*

  It shows the worker assignment to workstation and job as scheduled by the DPS module.

- *Wstation*

  When activated, it generates workstation's layout using six sub-menus: *A. Assign*, *H. Assign*, *Forecast*, *SpeedUp*, *SlowDown*, *Help*, and *Quit*. Such a layout is shown in Figure 3. The user can view changes in workstation status by clicking *Simulation* button. It allows the user to change worker assignment if *H. Assign* item is selected. The command items are detailed as below:

  - *A. Assign*

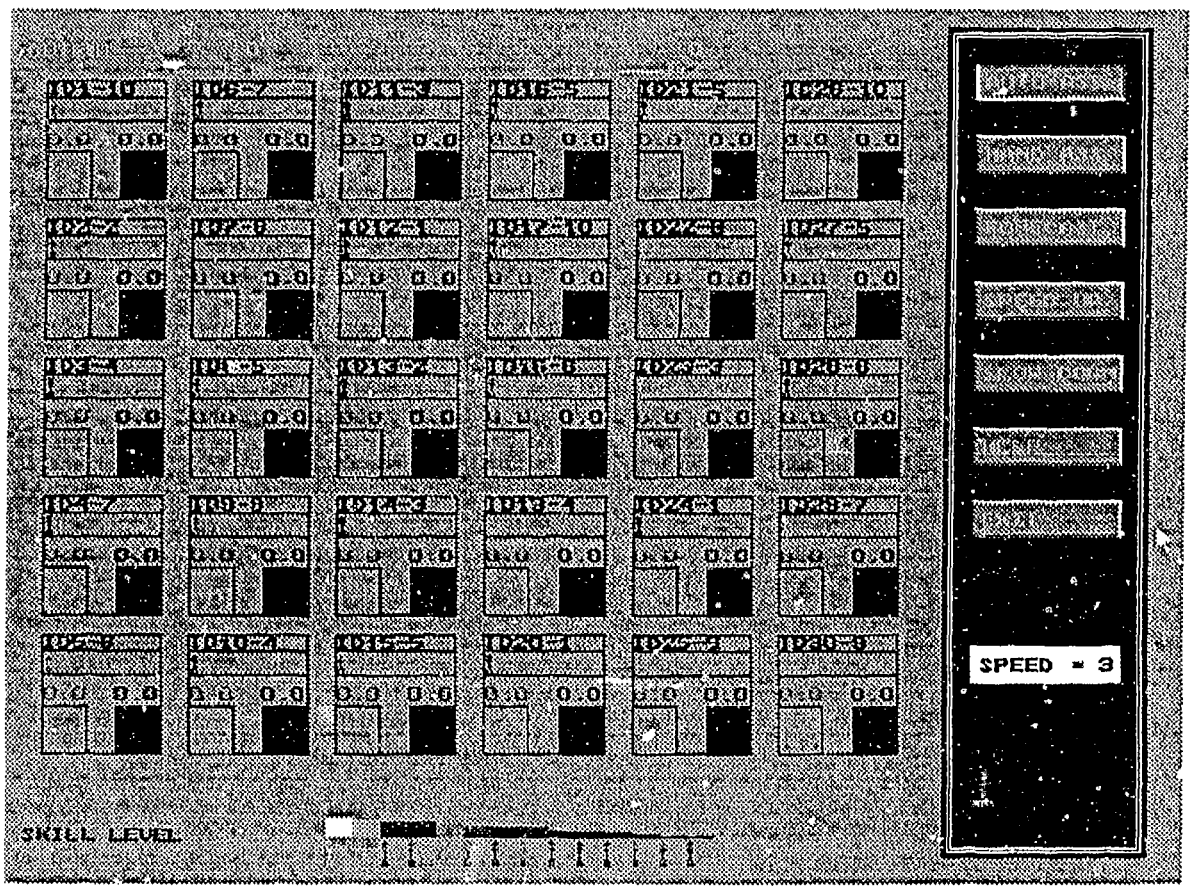
    It coordinates a simulation session in which the system automatically re-assign workers as a need is identified.

  - *H. Assign*

    It coordinates a simulation session which allows the user to relocate workers to a different workstation and updates all changes during the process. For example, when a worker is moved to a new station, the efficiency may change due to a different operation type. When all workers in a workstation are idle, the color of the workstation will turn into red to alert the production manager to interact with the system and assign workers to elsewhere. Thus the worker's record is partially modified and the worker's representation on the screen is updated, and the simulation will be run under the new worker assignment.

  - *Forecast*

    It triggers a simulation session and shows the physical layout of workstations, worker assignment to workstation, and worker assignment to job. It also shows worker skill level at the assigned workstation and the dynamic change of WIPs and the color of workers' icon representing the current skill level of each worker. The raw materials and finished good inventory level for the simulation session are also shown on the screen.


SPEED = 3
SKILL LEVEL

••• *Quit*

It returns the control to its parent menu.

•• *Tree*

When activated, it shows the operation tree for each job. Once selected, the user is asked to input the job ID for which the logical layout to be viewed. The menu consists of the following sub-menus:

••• *Up*

It allows the user to move up the logical layout of a operation tree for a job.

••• *Down*

It allows the user to move down the layout.

••• *Left*

It moves the layout to the left.

••• *Right*

It moves the layout to the right.

••• *Larger*

It enlarges the layout.

••• *Smaller*

It scales down the layout.

••• *Text*

It allows the user select and workstation in the layout to see a detailed textual description of the workstation. When a workstation is clicked, a text screen pops up with the following workstation related data: 1) workstation ID, 2) the operation type assigned to the workstation, 3) workers' ID and name, 4) current sewing efficiency at this station, 5) job IDs, and 6) bundle ID & sizes. The text window is removed and the layout is resumed by pressing the ESC key.

••• *Zoom*

The command item allows the user to zoom in a selected workstation. By clicking this item and then a workstation icon in the layout on the screen, the selected workstation is enlarged to better see the details of the station. The station is scaled back by pressing the ESC key.

••• *Forecast*

It starts a simulation session with the current layout and up to the

moment data. It shows the layout, worker to workstation assignment, and worker to job assignment. It also shows worker's skill level at each assigned workstation and the time-phased changes in WIP levels and worker assignment at each station.

••• *Quit*

It returns the control back to its parent menu.

•• *Quit*

It returns the control back to the menu.

3.2.6 *Exit* Menu

This control item has not sub-menu. The only command is *Exit*, which ends the system session and returns the control to the Microsoft Window.

APPENDIX A2: SOFTWARE REQUIREMENTS SPECIFICATION

SOFTWARE REQUIREMENTS SPECIFICATION

FOR THE APPAREL PRODUCTION PLANNING AND SCHEDULING SYSTEM
developed as part of the project entitled

A STUDY OF THE PRODUCTION PLANNING AND SCHEDULING PROBLEM IN THE APPAREL MANUFACTURING INDUSTRY
PHASE II
(#DLA900-91-C-1481)

submitted to

Defense Logistics Agency
Manufacturing Engineering Branch
Cameron Station-DLA-PRM
Alexandria, Virginia 22304-6100
Attn: Mr. Dan Gearing

by

Dr. Chin-Sheng Chen
Industrial & Systems Engineering Department
Florida International University
University Park, ECS 416
Miami, Florida 33199
Tel: (305) 348-3753

March 31, 1995

TABLE OF CONTENTS

1. GENERAL DESCRIPTION

1.1 Purpose

This Software Requirements Specification for the apparel production planning and scheduling system (APPSS), developed at Florida International University on the research project entitled, "A Study of the Production Planning and Scheduling Problem in the Apparel Manufacturing Industry" under the project number (DLA900-91-C-1481), is written to provide:

a. a base for mutual understanding between the developer and the customer, and

b. a basis for the development of software system tests.

1.2 Project Reference

"A Study of the Production Planning and Scheduling Problem in the Apparel Manufacturing Industry," a final technical report, (Contract #: DLA900-91-C-1481), Defense Logistics Agency, March, 1995.

2. SYSTEM SUMMARY

This chapter provides a summary of the system. The background is given in the following section, followed by a description of the objective in section 2.2 and a definition of the system in section 2.3. An identification of computer programs is provided in section 2.4. Major assumptions and constraints for the system are listed in section 2.5.

2.1 Background

Most apparel manufacturers, especially small and medium sized companies, engage in make-to-order production. The production planning and scheduling practice of a make-to-order is quite different from make-to-stock production because the survival of such a company depends solely on its flexibility and ability to quickly respond to an order inquiry. When an order is being considered, two important questions need to be promptly answered. They are 1) the likelihood of meeting the delivery date and 2) the manufacturing cost incurred by the order.

The domestic apparel manufacturing industry has been facing stiff competition from its overseas counterparts, the apparel industry is fighting an uphill battle. One factor which will work toward the advantage of domestic apparel manufacturers is their proximity to quickly respond to the changes in the market place. In

fact, quick response and flexibility have been considered an important strategy to the survival of this industry. Many efforts such as unit production, modular manufacturing, and CIM applications to this industry are aimed at achieving this goal.

The application of quick response concept requires a "quick" system to estimate the manufacturing lead time of potential order, develop an accurate production plan and schedule, acquire production resources. manufacture, and deliver the order promptly. At this preliminary decision-making stage of accepting an order, however, it is difficult to accurately generate a detailed production plan, manning schedule, and estimate the cost to meet a delivery date. The level of difficulty increase in a situation where large increase in production is required over a short period of time and the company must substantially increase hiring and training activities. This situation commonly occurs when a small apparel manufacturer accepts a contract to produce military apparel. Without accurate production plans and estimates, severe delay in delivery and excessive manufacturing cost may result.

There are several factors contributing to this difficultly. Delays on the materials delivery, high employee turnover, and static production planning methodology are common problems. In additions to these, an accurate estimate of the production capacity is another serious problem. Since the ability to change styles quickly is important to a make-to-order company, it has to rely on general-purpose machines to maintain its flexibility. Therefore extensive hiring, training, and retraining activities are required whenever there is a significant change in garment design or in production quantity. It is hard to measure production capacity in this type of environment. Also the limit of training capacity and the learning effect on production capacity over time are frequently underestimated.

Currently the production planning and scheduling practices do not fully address the impacts of production sequence and training effects on production capacity. These factors usually make a significant contribution to the delay and cost increase. Take the example of balancing a garment production line. Conceptually balancing a garment production line is quite simple. We determine what tasks are required to completely assemble one unit of garment and then divide up the tasks among the workers nearly equally so that all of the tasks get done. Because the work is nearly equally divided among the workers there is little worker idle time. And therefore theoretically it is a good approach. It is a good approach only on the assumption that there is no learning involved and every worker operates at a constant efficiency level. It can not be applied in a new situation where a great amount of training and retraining activity is required to cope with dramatic changes in garment styles or a sharp increase in production volume in a short period of time.

Apparel manufacturing firms frequently have trouble meeting delivery commitments for finished products. The trouble may be caused by unexpected absenteeism, equipment breakdown, or delay in receiving raw materials. To the apparel manufacturing industry, these are common situations. Most managers in well run firms are able to establish production schedules which allow for these factors. A firm can use past experience to produce a fairly reliable production schedule if the firm accepts an order for a product which is similar in style to its current production, and the size of the order and delivery date are such that the current work force can handle the production.

In situations where there is a radical change in style or there is a requirement for a significant increase in production over a short time period, even well managed companies frequently have difficulty in establishing an accurate production schedule. These two situations require extensive training or retraining of machine operators. Apparel manufacturing firms do not have an effective way of incorporating the effects of these training production capacity is quite significant. For example, it may rake more than 16 weeks for a new fire to reach 100% efficiency on some sewing operations, requiring a high degree of skill.

Realistically, in a production line manned with a high percentage of operators with low proficiency, a momentary balance due to the difference in learning speed of each operation. In order to solve this problem, one notion could be re-balance it again by shifting the operators around. This will not solve the problem because it again will require training, and the operators being shifted will have to start with very low proficiency. On the contrary, it complicates the problem, causes further delay and costs more. For such a situation, a new look at the line balancing is necessary.

Basically what we have discussed are production planning and scheduling problem, although not traditional ones. To solve these problems, there is a need to investigating the relationships and interactions among those actors contributing to these problems, develop a methodology, and implement it in a computer as a planing and scheduling system. This system, taking into account the factors discussed above, would be a tool for quickly estimating the feasibility of meeting a delivery deadline and manufacturing cost in response to an order inquiry at the stage accepting an order. It would be a tool for generating a realistic production plan and measure the level of reliability of this plan at the stage of production planning and scheduling. Due to the interactive nature of these factors, these problems must be treated and solved as an integrated problem. When developing a solution to this integrated planning problem, one way of doing it is dividing this problem into a number of smaller problems, solving them separately, and then piecing them together again.

2.2 Objectives

This project proposed to develop a solution technique for the production planning and scheduling problem in the apparel manufacturing industry, to maximize the profit while meeting job due date and other constraints. The technique was implemented into a hierarchical production planning and scheduling system which performs the following functions:

1) manage apparel product, production, and manufacturing information.
2) evaluate the feasibility of meeting delivery commitments.
3) select the best production method.
4) generate a master production schedule.
5) generate a detailed production schedule
6) evaluate the static production plans.

2.3 System Definition

The architecture of the proposed system is shown in Figure 1. It contains the following modules: a graphic user interface, a database manager, a production method evaluator, a sewing skill predictor, a capacity planner, a master production scheduler, and detailed production scheduler, and a material flow simulator.

The system relies on many apparel product, production, and manufacturing data. The database manager manages these data and provides typical database functionalities to operate the system such as browse, search, add, edit, delete, print. The master production scheduler generates a weekly production plans and can be used to evaluate the feasibility of meeting the delivery date of a new order. If it can not be schedule on the master production plan within the due date, the order should not be accepted. The production method evaluation module evaluates a variety of production methods for a given order. It provides the input for the user to select the best production method. However, the current system is focused on the bundle production method.

During master production scheduling, the sewing skill prediction module is routinely called to estimate worker's skill proficiency at the assigned operation type and the plant capacity. When the current capacity cannot meet the production, the capacity planning module is activated to select additional short term resources. The detailed production scheduling module takes the master production schedule and other required data from the database and generates a detailed production schedule. The material flow simulation module is then used to validate the production plan to further ensure its validity under a dynamic production environment. The simulation module retrieves the most up to date data from the database and allow the user to re-assign workers to different stations to smooth the material flow.

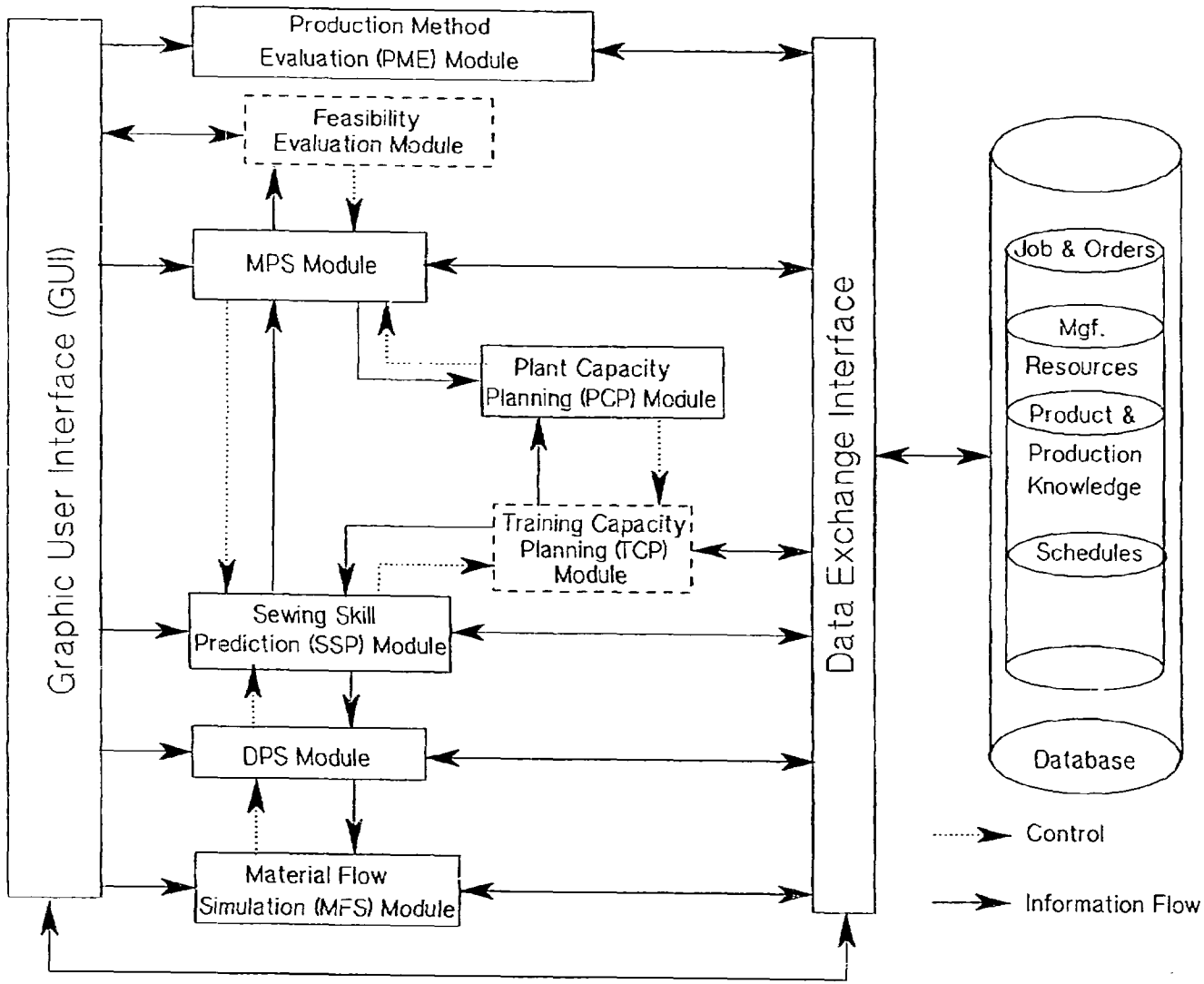


Figure 1: Proposed Apparel Production Planning & Scheduling System Architecture

2.4 Computer Program Identification

This section lists major computer programs implemented in this study to implement the proposed system. Each of them is assigned with a unique identifier in this report, which is a sequence of numbers and letters indicating the module it is created for.

2.4.1 Programs for the master production scheduler

1MPS01: CommitedSubQty
2MPS02: FinishedQty
3MPS03: GetAllOptype
4MPS04: GetAllWorker
5MPS05: GetCurrWeek
6MPS06: GetJob
7MPS07: GetStyle
8MPS08: IsOverTime
9MPS9: Mps
10MPS10: RecoverFiniQty
11MPS11: SelectStime
12MPS12: SelectWorker
13MPS13: Umps
14MPS14: UpdateCurrTrain
15MPS15: UpdateFiniQty
16MPS16: UpdateThisTrain
17MPS17: UpdateUmpsWk
18MPS18: UpdateUmpsDB

2.4.2 Programs for the detailed production scheduler

19DPS01: GetOperId
20DPS02: IniJobTask
21DPS03: IniUpperBound
22DPS04: InputData
23DPS04: JobIndex
24DPS06 OperationIndex
25DPS07 OperationTypeIndex
26DPS10: ToDpsJWk
27DPS11: WorkerIndex
28DPS12: WorkerRank
29DPS13: WorkerAssignJob
30DPS14: WorkerAssignOp

2.4.3 Programs for the simulator

31SC01: JobTree
32SC02: OneWorker
33SC05: ShowOperation
34SC06: ShowOneOperation
35SC07: ShowTree
36SC10: SimMenu

37SC11: Simulation
38SC12: TextOper
39SC13: WeekTask
40SC14: WhichWorker
41SC15: WorkerPost
42SC16: ZoomOper

2.4.4 Programs for the skill prediction module

43SKILL01: NonLinearFun

2.5 System Assumptions and Constraints

The following major assumptions and constraints are identified for the system.

1) Only bundle production method is considered in this study.

2) The apparel manufacturing firm is engaged in assemble-to-order production. In other words, no cutting, pre-assembly, and assembly production activities are used to smooth production levels.

4) Job orders are relatively small and thus short prodution runs are the norm.

5) General purpose machines are used. Production resources such as sewing machines are assumed sufficient but the plant capacity depends on the sewing workers.

6) Workers have multiple sewing skills. However, the skill at each operation type varies, depending on the amount of training received for each operation type.

7) The capacity and efficiency of one workstation thus depends on the nominal capacity of the station and more importantly the sewing skill of individual workers assigned to the station.

8) The management may rely on frequent re-assignment of workers to different workstations to smooth the material flow.

9) Jobs and orders may be accepted at any time and thus be scheduled at any time.

10) Each job is assigned with a due date and penalty as set by the customer during ordering. No penalty for early finishing and delivery.

11) Each job is known of its operations, standard time, and operation sequence.

12) Plant capacity is know in terms of the momentary skills of each worker at each operation type. The capacity of additional resources such as subcontracting is also known in the same fashion.

13) A workstation may have a number of machines and can accommodate a number of workers. Several jobs may be routed to one workstation. But each workstation is assigned to perform only one operation type at any given time.

14) A job may visit workstations in any predetermined sequence, as many stations as the number of the operations. But no job is allowed to visit a workstation more than once.

3. SYSTEM ENVIRONMENT

This chapter is organized into four sections. The required equipment environment is defined in the next section, followed by a description of the support software environment in section 3.2. The interface and the data security issues are detailed in sections 3.3 and 3.4.

3.1 Equipment Environment

The operations of this system requires the following hardware as described below:

Hardware	Description/Comments
Computer	PC/386 or higher.
RAM	4MB.
Hard disk	30MB at minimum.
Video monitor	VGA or higher.
Mouse	required.
Printer	required.

3.2 Support Software Environment

This system also requires support of the following software packages.

Software Packages	Description
Microsoft Windows	Version 3.1 or later.
Data Management System	Paradox/DOS Windows 4.5 or later.
Data Exchange tool	Paradox Engine 3.0 or later.
Programming Language	Turbo C++ 3.0.

3.3 Interfaces

This section describes the user interfaces provided with the system. As shown in Figure 1, the user interfaces with this system through the following interface designs:

Inter01: interface with the database;
Inter02: interface with the production method evaluator;
Inter03: interface with the feasibility evaluation module;
Inter04: interface with the master production scheduler;
Inter05: interface with the sewing skill predictor;
Inter06: interface with the detailed production scheduler;
Inter07: interface with the material flow simulator;.

3.4 Security and Privacy

This section specifies the input and output of each system module and the security and privacy restrictions associated with the data being handled.

The database is the heart of the system. All the modules need to retrieve data from the database; and in turn they also update the database, change some fields of some tables in the database. The information flow may also occur between modules. In the data exchange process, some conflict may happen if data was manipulated improperly. Some privacy restrictions must be provided. They are summarized in the following subsections.

3.4.1 Database

The database was designed physically with several tables which can be classified by Order (ORDER, JOB), Resources (OPER, WORKER, WS, WK_OPTP), Knowledge (MISCELL, OPTYPE, STIME, STYLE, STYLE_OP), and Schedule (DPS_J_WK, DPS_WIP, UMPS, UMPS_WK).

3.4.1.1 order related data

1) ORDER: (Order_id, Customer_id, Order_date)

This table stores customers' orders. It stores order ID, customer ID, and order arrival date. These three attributes decide one unique order.

Restriction:
No restriction is applied to this table.

2) JOB: (Job_id, Order_id, Style_id, Due_date, Due_week, Penalty, Color_id, Size_id, Quantity, Finished_qty, Dozen_num, Unit_price, Status, Cost, Subcontract, Acceptability, Committed_subcontract_qty, Ini_finished_qty)

This table captures the information about jobs. The attributes of a job include its ID, the order it belongs to, the style ID indicating the tree structure of operations for making this garment, the due date assigned to the job, the due week, translated from the due date, penalty, color ID, garment size, quantity, finished quantity (quantity finished up to the current week), bundle size, unit price, job status (new, finished, in process), manufacturing cost, the maximum amount of subcontract the job is allowed, the committed amount of subcontract, and the acceptability of the job.

Restrictions:

Each job has a unique job ID in an order. It is given automatically by system. *Due_week* must be later than the *Curr_week* in *MISCELL* table. *Status* and *Acceptability* can not be edited by the user. It can be updated by the master production scheduler, usually done once a week.

3.4.1.2. resources related data

1) OPER: (Operation_id, Optype_id)

This table stores the operations and their operation type. All the possible operations of the apparel plant are defined in this table.

Restrictions:

Operation_id is the key. It is generated by the computer program. It can not be changed by the user. The *Optype_id* must be one of the same as the one in the *OPTYPE* table.

2) WORKER: (Worker_id, Name, Hire_date, Overtime1, Overime2, Overtime3, Overtime4, Overtim5, Overtime6, Overtime7, Overtime8, Overtime9, Overtime10, Absence1, Abcence2, Abscence3, Abscence4, Absence5)

This table captures the attributes of workers. The attributes are worker's ID, the worker's name, his/her hire date, and a maximum of ten days available for overtime, a maximum of five planned absence days.

Restrictions:

Every worker has an unique *Worker_id*, which is the key. It is generated by a computer program. All the attributes except *Worker_id* can be added and edited by the user. The number of characters in the *Name* must be less than eight. The overtime and absenteeism can be retrieved only by the capacity planning module.

3) WK_OPTP: (Worker_id, Optype_id, Ini_efficency, Ini_train_days, Curr_efficiency, Curr_train_days,

This_train_status)

The skill level of a sewing worker may be different at a different operation type. It is assumed to be dependent on the number of training days at the operation type. This table stores the worker's initial training days (when the worker was hired), the initial efficiency, the current training days, the current efficiency, and this week's training status at a given operation type.

Restrictions:

The combination of *Worker_id* and *Optype_id* is the key. All the attributes except *This_train_status* in this table can be added and edited by the user. This_train_status can be changed only by the master production scheduling module and the material flow simulator.

4) WS: (Wkstation_id, Optype_id, Max_workers, Max_inventory, Position_x, Position_y)

The table stores the workstation ID, optype ID, maximum production capacity of hte workstation, maximum inventory level, position x, and position y. The positions x and y are used to define the workstation's location in the layout.

Restrictions:

The workstation_id is the key. All attributes except workstation_id in the table can be added and edited by the user.

3.4.1.3 production knowledge

1) MISCELL: (Application_levle, Curr_week, Days_in_week, Ini_day, Overtime, Newhire, subcontract, days_in_week, Hours_in_day)

This table captures miscellaneous information required for the operations of the system. The information do not belong to any entity. *Current_week* records the current week according to the system calendar. *Days_in_week* keeps track of the number of working days in a week, which is generally five days. *Hours_in_day* denotes the number of work hours each day. It is assumed to have eight work hours a day for a single shift operation.

Restrictions:

No key is used in this table. The *Application_level* has only two values: 0 and 1. Only the Master production scheduler can change the value of *Curr_week* when the value of *Application_level* is 1. When it is set to 0, all the attributes in this table can be edited by the user.

2) OPTYPE: (Optype_id, Name, F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, F13, F14, F15, F16, F17, A, B, C, Unit_cost)

This table captures all possible operation types which the apparel plant can do. For each operation type, it has a name, seventeen operation variables (F1-F17), and three predictors (A, B, C), which are used by the skill prediction module to estimate worker skill at an operation at any given time.

Restrictions:

Every operation type has an unique *Optype_id* and *Name*. The *Optype_id* is the key. The operation variables and skill predictors can only be retrieved by the sewing skill predictor. All attributes except *Optype_id* in this table can be edited by user. It is automatically generated by the computer program.

3) STIME: (Style_id, Size, Op_id, Standard_time)

Standard processing time can be decided only when the style and the garment size are given. This table stores the standard processing time of all combinations of style, size, and operations.

Restrictions:

The combination of *Style_id*, *Size_id*, and *Op_id* is used as the key of this table. These three attributes can be edited. *Standard_time* can be retrieved by the master production scheduler, the detailed production scheduler module, and the material flow simulation module.

4) STYLE: (Style_id, Root_operation_id)

Style is an attribute of jobs. It captures the root operation of the garment style and can be used to retrieve all other operations required for this style. The operation structure and sequence constitute a style and represented by the root operation. Once the root operation ID is given, all operations and their sequence can be found through the table *STYLE_OP*.

Restrictions:

Style_id is the key and is generated by the computer program. The *Root_operation_id* can be added or edited by the user.

5) STYLE_OP: (Style_id, Operation_id, Prev_op_id1, Prev_op_id2, Prev_op_id3, Next_op_id)

For every combination of operation and style, there exists a number of preceding operations and one subsequent operation. This means that several components are allowed to be assembled into one sub-assembly or assembly. The maximum number of preceding operations is currently limited by three in this system.

Restriction:

The combination of *Style_id* and *Operation_id* is the key. All values of the attributes can be added and edited by the user. *Prev_op_id1*, *Prev_id2*, *Prev_id3*, and *Next_op_id* can be retrieved by the master production scheduler and the detailed production schedule module

3.4.1.4 schedules

1) DPS_J_WK: (Day_id, Wk_id, Optype_id, Ws_id, Job_id)

This table stores the detailed production schedule and its attributes of worker ID, operation type, workstation ID, and job ID, and the expected daily throughput of this worker.

Restrictions:

The combination of *Day_id*, *Wk_id*, *Optype_id*, *Ws_id*, and *Job_id* is the key. All the attributes are updated by the detailed production scheduling module or the material flow simulation module. Throughput can be read and edited by the user and the simulation module.

2) DPS_WIP: (Day_id, Ws_id, Job_id, Wip_in, Wip_out, Wip_inventory)

This table captures a part of the detailed production schedule indicating the total WIP inventory at beginning and at the end of each day by workstation and by job.

Restrictions:

The combination of *Day_id*, *Ws_is*, and *Job_id* is the key. All attributes are updated by the detailed production scheduling module and the simulator. The user can edit *Wip_in*, *Wip_out*, and *Wip_inventory*.

3) UMPS: (Week_id, Job_id, Throughput)

This table stores the master production schedule, which stores the planned throughput of each job in each week within the planning horizon.

Restrictions:

The combination of *Week_id* and *Job_id* is the key. All attributes are updated by the master production scheduling

module. They can also be retrieved by the user and the detailed production scheduler.

4) UMPS_WK: (Week_id, Worker_id, Optype_id)

This table also stores the master production schedule which captures the worker assignment by operation type.

Restrictions:
The combination of *Week_id*, *Worker_id* and *Optype_id* is the key. Al attributes are updated by the mater production scheduling module. They can not be edited by the user and other application modules.

3.4.2 Master production scheduling module

1) Input data:
Curr_week , Days_in_week, Hours_in_day from MISCELL table; Job_id, Style_id, Due_week, Penalty, Size_id, Quantity, Finished_qty, Dozen_num, Unit_price, Status, Cost from JOB table; Worker_id from WORKER table; Standard_time from STIME table; Operation_id from OPER table; Optype_id from OPTYPE table; Prev_op_id1, Prev_op_id2, Prev_op_id3, Next_op_id from STYLE_OP table; Curr_efficency from WK_OPTP table.

2) Output data:
Status, Finished_qty, Acceptabilit[illegible] JOB table; Curr_week to MISCELL table; Curr_train_days, this_train_status to WK_OPTP table; Day_id, Wk_id, Optype_id, Ws_id, Job_id, Throughput to DPS_J_WK table; Week_id, Job_id, Throughput to UMPS table; Week_id, Worker_id, Optype_id to UMPS_WK table.

3.4.3 Detailed production scheduling module

1) Input data:
Curr_week Days_in_week, Hours_in_day from MISCELL table; Job_id, Throughput from UMPS table; Dozen_num, Style_id, Size_id from JOB table; Root_operation_id from STYLE table; Prev_op_id1, Prev_op_id2, Prev_op_id3, Next_op_id from STYLE_OP table; Operation_id from OPER table; Worker_id, Optype_id from UMPS_WK table; Standard_time from STIME table; Ini_efficiency, Curr_effi ency from WK_OPTP table.

2) Output data:
Day_id, Wk_id, Optype_id, Ws_id, Job_id, Throughput to DPS_J_WK table; Day_id, Ws_id, Job_id, Wip_in, Wip_out, Wip_inventory to DPS_WIP table.

3.4.4 Sewing skill prediction module

1) Input data:

Ini_train_days, Curr_train_days, F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, F13, F14, F15, F16, F17, A, B, C from OPTYPE table.

2) Output data:
Ini_efficiency, Curr_efficiency to WK_OPTP table.

3.4.5 Plant capacity planning module

1) Input data
Curr_week from MICELL table; Subcontract from JOB table; Overtime1, Overime2, Overtime3, Overtime4, Overtim5, Overtime6, Overtime7, Overtime8, Overtime9, Overtime10, Absence1, Abcence2, Abscence3, Abscence4, Absence5 from WORKER table.

2) Output data:
Finished_qty, Committed_sub, Acceptability, Status to JOB table.

4. DETAILED CHARACTERISTICS AND REQUIREMENTS

This chapter defines and specifies system performance, functional and operational requirements, design constraints and standards necessary to ensure proper development and maintenance of the system.

4.1 Specific Performance Requirements

This section is used to describe the specific performance requirements to be satisfied by the system. There is no specific performance requirement defined for this system. The system can carry out each application module within a reasonable computer time. The skill prediction model can estimate skill efficiency within 10%, according to the test data collected.

4.2 Computer Program Functions

This section describes the individual functional requirement of each program identified in Section 2.5. Under each application module, each program is identified with its function name, a program description, a syntax description, and remarks including its functional and the performance requirement, and its relationship with other functions.

4.2.1. Master production scheduling functions (1-18)

1. CommitedSubQty

Description Writes a integer value to JOB table.

Category MPS function
Syntax void CommitedSubQty(int job_id, int quantity);
Remarks Updates Committed_sub field of the record which the value of Job_id is equal to job_id in JOB table, no return value. It is a sub-function of Mps.

2. FinishedQty

Description Preserves Finished_qty of JOB table
Category MPS function
Syntax void FinishedQty(int job_id[], int job_num);
Remarks This function searches for the jobs which the value of Job_id is equal to job_id, preserves the value of Finished_qty into ini_finished_qty, so that when the value of Finished_qty is updated by UpdateFiniQty(), its value still can be retrieved from Ini_finished_qty. No return value. It is a sub-function of Mps.

3. GetAllOptype

Description Retrieves all operation types from OPTYPE table
Category MPS function
Syntax
```
#include "optype.h"
int  GetAllOptype(Optype *& all_optype)
```
Remarks GetAllOptype opens the OPTYPE table, retrieves the value of all the record into an array which is pointed by all_optype.
Optype is a C++ class which is defined in optype.h:

```
class     Optype{
private:
     int        optype_id;
     double     a;
     double     b;
     double          c;
     double     f[17];
public:
     Optype();
     ~Optype();

     void       SetId(int a_optype_id);
     int        GetId();
     void       Seta(double a_a);
     double          Geta();
     void       Setb(double a_b);
     double          Getb();
     void       Setc(double a_c);
     double     Getc();
};
```

The number of record retrieved is returned, if this procedure is failed the return value is less than

zero. This function is a sub-function of Mps.

4. GetAllWorker

Description Retrieves all available workers from WCRKER ta' le
Category MPS function
Syntax
```
#include "worker.h"
int GetAllWorker(Worker*& all_worker);
```
Remarks This function opens the WORKER table, retrieves all the record into an array which is pointed by all_worker.
Worker is a C++ class which is defined in worker.h:

```
class     Worker{
private:
int  worker_id;
int     training_days[MAX_NUM_OPTYPE];
int  last_optype_id;
int  curr_optype_id;
int     overt[MAX_NUM_OVERTIME];
int     abse[MAX_NUM_ABSE];
int  avail;
public:
Worker(int a_worker_id);
Worker();
~Worker();
void SetId(int a_worker_id);
int  GetId();
void SetTrainningDays(int a_optype, int a_days);
int  GetTrainningDays(int a_optype);
void SetLastOptypeId(int a_last_optype_id);
int  GetLastOptypeId();
void SetCurrOptypeId(int a_curr_optype_id);
int  GetCurrOptypeId();
void      SetOvert(int a_index, int a_week);
int         GetOvert(int a_index);
void       SetAbse(int a_index, int a_week);
int         GetAbse(int a_index);
void SetAvail(int a_avail);
int  GetAvail();
double    OptypeEff(int optype_id, Optype optype[]);
};
```

This function returns the number of worker retrieved, if the procedure fails, it returns a value less than zero. It is a sub-function of Mps.

5. GetCurrWeek

Description Retrieves a integer from MISCELI table
Category MPS function
Syntax
```
int  GetCurrWeek(void);
```
Remarks This function retrieves the value of Curr_week from

MISCELL. Return the value of Curr_week. It is a sub-function of Mps.

6. GetJob

Description Retrieves all jobs' information from JOB table
Category MPS function
Syntax

```
#include "job.h"
Job&  GetJob(int job_id[]);
```

Remarks GetJob opens the JOB table, searches for the jobs which the Job_id equals to job_id, returns the value of Style_id, Due_week, Size_id, Quantity, Finished_qty, Subcontract, Dozen_num. It is a sub-function of Mps. Job is a class, the structure :

```
class     Job{
private:
int  job_id;
Style*     style;
int  due_week;
int  size_id;
int  quantity;
int  finished_qty;
int          sub_qty;
int  dozen_num;
int  status;
public:
Job();
~Job();
void      SetJobId(int a_job_id);
int  GetJobId();
void SetStyle(Style*);
void      GetStyle(Style*&);
void SetDueWeek(int a_due_week);
int  GetDueWeek();
void SetSizeId(int a_style_id);
int  GetSizeId();
void SetQty(int a_quantity);
int  GetQty();
void SetFinishedQty(int a_finished_qty);
int  GetFinishedQty();
void      SetSubQty(int a_sub_qty);
int         GetSubQty();
void SetDozenNum(int a_dozen_num);
int  GetDozenNum();
void SetStatus(int a_status);
int  GetStatus();

void SetStime();
int         Decomposition(int qty, double stime[]);
};
```

7. GetStyle

Description Retrieves styles from STYLE table
Category MPS function
Syntax
```
#include "style.h"
Style& GetStyle(int style_id);
```
Remarks GetStyle opens the STYLE table, searches for the style that the Style_id equals to style_id, then returns all the information of the style, including Prev_op_id1, Prev_op_id2, Prev_op_id3, Next_op_id. It is a sub-function of Mps. Style is a class which has a subclass Opstyle, their structure are shown:

```
class Style {
private:
     int        style_id;
     OpStyle    *root;
     OpStyle    *op_style_array;
     int        op_style_num;

public:
     Style();
     Style(int a_style_id);
     ~Style();
     void SetId(int a_style_id);
     int  GetId();
     void SetRoot(OpStyle*);
     void GetRoot(OpStyle*&);
     void SetNumOpStyle(int);
     int  GetNumOpStyle();
     void SetOpStyleArray(OpStyle*);
     OpStyle* GetOpStyleArray();
     void MidTravel(OpStyle*  a_opstyle,  OpStyle
opstyle[], int& n);
     void AllOptype(int typelist[], int& n);
};



class OpStyle {
private:
     int        op_id;
     OpStyle         *prev1, *prev2, *prev3;
     OpStyle    *next;
     float           stime;

public:
     OpStyle();
     OpStyle(int a_op_id);
     ~OpStyle();
```

```
        void SetId(int a_op_id);
        int  GetId();
        void      SetStime(double);
        double    GetStime();
        void   SetPrev(OpStyle*   a_prev1,   OpStyle*
        a_prev2, OpStyle*a_prev3);
        void GetPrev(OpStyle*&   a_prev1,   OpStyle*&
        a_prev2, OpStyle*& a_prev3);
        void      SetNext(OpStyle* a_next);
        void      GetNext(OpStyle*& a_next);
};
```

8. IsOverTime

Description Finds out one worker's overtime or absent status

Category MPS function

Syntax
```
#include "worker.h"
int  IsOverTime(WORKER worker[], int wn, int wk_id,
int curr_week);
```

Remarks Parameter worker is an array which stores wn number of WORKERs (refer to GetAllWorker for the structure of WORKER). Three return values: -1, indicates this worker is absent this week; 0 indicates this worker is neither absent nor overtime; 1, indicates this worker is overtime this week. IsOverTime is a sub-function of Mps().

9. Mps

Description Creates master production schedule under a certain capacity level

Category MPS function

Syntax
```
#include "worker.h"
#include "style.h"
#include "job.h"
#include "optype.h"
#include "pcengine.h"
int  Mps(int capacity_level,  int curr_week);
```

Remarks Parameter capacity_level has three values: 0,1,2. They represent three capacity level in current week.

0: normal capacity(no overtime, subcontract, no newhire),
1: overtime and subcontract only,
2: all three alternatives are available.

After this function is called, the UMPS, UMPS_WK, WORKER, MISCELL, WK_OPTYPE, JOB tables are updated. This function is a sub-function of main function of MPS module.

10. RecoverFiniQty

Description Recover the updated Finished_qty in JOB table
Category MPS function
Syntax
```
#include "pcengine.h"
void  RecoverFiniQty(int job_id[], int jn);
```
Remarks Parameter is an array which stores the job IDs that the Finished_qty need to be recovered, jn is the job number. This function opens the JOB table, allocates the jobs, restores the value of Ini_finished_qty to Finished_qty, please refer to the function FinishedQty.
RecoverFiniQty is sub-function of Mps, no return value.

11. SelectStime

Description Pick the most urgent operation type
Category MPS function element
Syntax
```
int  SelectStime(double stime[], int sn);
```
Remarks SelectStime searches for the largest value in array stime which has sn **double** elements, and returns the index. This function is a subfunction of Umps.

12. SelectWorker

Description Pick the most efficient worker
Category MPS function
Syntax
```
#include "worker.h"
int   SelectWorker(Worker worker[], int wn, int
curr_week);
```
Remarks SelectWorker searches for the most efficient worker value in array worker which has wn **Worker** elements, and returns the index. Refer to GetAllWorker function for the definition of class **Worker**. SelectWorker is a sub-function of Umps.

13. Umps

Description Assigns workers to each operation type
Category MPS function
Syntax
```
#include "worker.h"
#include "optype.h"
int Umps(double stime[], int sn, Worker worker[],
int wn, Optype optype[], int capacity_level, int
curr_week)
```
Remarks The average quantity that should be finished in current week is transferred to standard time of each operation type which is stored in an array stime, sn is the number of elements in the array which is the number of operation type. All available worker is stored in another array(worker) that has wn elements, the information of each

operation type is stored in the array optype, the capacity level of current week is also need to be set. Umps tries to assign available workers to each operation type to see if available workers can finish required standard processing time in current week under the capacity levle. Umps returns -1 when some stime elements are not zero but no available workers left, in this case, the estimated standard processing time is more than the standard time that workers can finish . Umps returns 1 when all the stime elements are equal to zero, but some workers are still available, in this case, the estimated standard time is too small. Umps is a sub-function of Mps.

14. UpdateCurrTrain

Description Update Curr_train_days in WK_OPTP table
Category MPS function
Syntax void UpdateCurrTrain(int curr_week);
Remarks UpdateCurrTrain opens the WK_OPTP table, updates the Curr_train_days according to Curr_train_days and This_train_status. If the worker is assigned to one operation type, the Curr_train_days of this worker will increase the value stored in Days_in_week in MICELL table one week later. No return value. UpdateCurrTrain is a sub-function of Umps.

15. UpdateFiniQty

Description Updates Finished_qty in JOB table
Category MPS function
Syntax void UpdateFiniQty(int job_id[], int jn, int qty[]);
Remarks UpdateFiniQty opens the JOB table, searches for the records that the value of Job_id equals to the value of the elements of array job_id, set the value of Finished_qty by the element of array qty. The total number of jobs to be updated is jn. No return value occurs. This function is a sub-function of Mps.

16. UpdateThisTrain

Description Update This_train_status in WK_OPTP table
Category MPS function
Syntax #include "worker.h"
void UpdateThisTrain(Worker worker[], int wn);
Remarks UpdateThisTrain opens the WK_OPTP table, if one worker is assigned to one operation type, the

This_train_status of this worker is set by UpdateThisTrain with 1, otherwise the status is set with 0. Refer to GetAllWorker for **Worker** class definition. No return value. UpdateThisTrain is a sub-function of Umps.

17. UpdateUmpsWk

Description Update the master production schedule into UMPS_WK table
Category MPS function
Syntax
```
#include "worker.h"
void UpdateUmpsWk(Worker worker[], int wn, int curr_week);
```

Remarks UpdateUmpsWk opens the UMPS_WK table, set Worker_id, Optype_id with the information stored in array worker. Refer to Worker class definition. wn is the number of worker. No return value. It is a sub-function of Umps.

18. UpdateUmpsDB

Description Update Job_id and Throughput of UMPS table
Category MPS function
Syntax
```
void UpdateUmpsDB(int job_id[], int jn, int qty[], int curr_week);
```
Remarks UpdateUmpsDB opens the UMPS, if the week indicated by curr_week already exists in the table, first, delete all the record. Then, append records, Week_id by curr_week, Job_id by one element of array job_id, Throughput by qty. No return value. UpdateUmpsDB is a sub-function of Umps.

4.2.2 Detailed production scheduling functions (19-30)

Following data structures must be defined to develop the functions of Detailed Production Schedule module and Simulation module:

```
# define MAX_JOB_NUM      4
# define MAX_OPTYPE_NUM   10
# define MAX_WK_NUM       100
//# define MAX_JOB_NUM    3
# define MAX_WJOB_NUM                4
# define MAX_JOB_BUNDLE        300
# define MAX_OP_WK_NUM    10
# define MAX_OP_JOB_WK_NUM      8
# define MAX_WEEK              16
# define ONCE_WEEK                   8
# define MAX_WKSTATION_NUM     30
# define ROW_OPER_NUM                6
# define COL_OPER_NUM                5
```

```
# define MAX_OPER_NUM                20
# define DAYS_IN_WEEK          5
# define HOURS_IN_DAY          8
# define WEEK                  20
# define SEGMENT               10
# define MAX_ASSEM             3
# define VIEW_X1               90
# define VIEW_Y1               100
# define VIEW_X2               470
# define VIEW_Y2               280
# define UP_X                        20
# define UP_Y                  40
# define BOTTOM_X              500
# define BOTTOM_Y              420
# define WP_UP_X                     50
# define WP_UP_Y                     60
# define WP_BOTTOM_X           500
# define WP_BOTTOM_Y           350
# define COLOR_START           15-MAX_WJOB_NUM
# define TREE_X                      50
# define TREE_Y                      25
# define TREE_H                      350
# define TREE_L                      400

struct _job{
        short   id;
        short        oper_num;
        short        oper_id[MAX_OPER_NUM];
        short   bundle_num;
        short   bundle_id[MAX_JOB_BUNDLE];
        double  st;
        short        ini_store[MAX_ASSEM];
        short   ass_num;
        short   store[MAX_ASSEM];
        short   doz_num;
        double  cost;
        short   avail;
};

struct _optype {
        short   id;
        short   wk_num;
        short   mps_wk_num;
        short   wk_id[18];
        short   wk_oper[18];
        short   oper_num;
        short   oper_id[MAX_OPER_NUM];
        short   remn_wk;
        short   remn_wk_id[10];
        short   remn_oper;
        short   remn_oper_id[MAX_OPER_NUM];
```

```
        short  avail;
        double predict_abc[3];
        double capacity;
};
struct _oper{
        short  id;
        short  optype;
        short  job_num;
        short  job_id[MAX_JOB_NUM];
        short  job_avail[MAX_JOB_NUM];
        double      max_iv[MAX_JOB_NUM];
        double t_max_iv;
        short  wk_num;
        short  wk_id[MAX_OP_WK_NUM];
        short  wk_job_num[MAX_JOB_NUM];
        short  wk_job_id[MAX_JOB_NUM][MAX_OP_JOB_WK_NUM];
        short  pre_oper_num[MAX_JOB_NUM];
        short  pre_oper_id[MAX_JOB_NUM][3];
        short  next_oper_id[MAX_JOB_NUM];
        int         store_index[MAX_JOB_NUM];
        short  x;
        short  y;
        double ini_iv_b[MAX_JOB_NUM];
        double job_rem[MAX_JOB_NUM];
        double task;
        double capacity;
        double job_task[MAX_JOB_NUM];
        double job_capacity[MAX_JOB_NUM];
        double      st[MAX_JOB_NUM];
        double      out_b_num[MAX_JOB_NUM];
        double      iv_b_num[MAX_JOB_NUM];
        double iv;
};
     struct _worker {
        short  id;
        short  urgent;
        short  avail[MAX_WEEK];
        double op_eff[10];
        short  ini_training_day[10];
        short  op_training_day[10];
        short  x;
        short  y;
        short  color;
};
 struct _job         job[MAX_JOB_NUM];
 struct _optype      optype[MAX_OPTYPE_NUM];
 struct _oper  oper[MAX_WKSTATION_NUM];
 struct _worker      worker[MAX_WK_NUM];
```

19. GetOperId

Description Find all operations for each operation type
Category DPS function
Syntax void GetOperId(void);
Remarks GetOperId initializes the oper_id in optype structure.

20. IniJobTask

Description Initialize the task of jobs
Category DPS function
Syntax void IniJobTask(void)
Remarks IniJobTask initializes the job_task in oper structure.

21. IniUpperBound

Description Initialize the inventory limitation
Category DPS function
Syntax void InitUpperBound(void)
Remarks IniUpperBound initializes the max_iv(maximum inventory) in oper structure.

22. InputData

Description Input data from database
Category DPS function
Syntax void InputData
Remarks InputData retrieve data from database, set the data into structures. The data need to be set is dozen_num, id, oper_num, cost, oper_id, ass_num, ini_store in job structure; id, predict_abc, mps_wk_num, wk_id, oper_num in optype structure; id, optype, job_num, job_id, pre_oper_id, next_oper_id, st, iv_b_num in oper structure; global variable job_num, oper_num, optype_num.

23. JobIndex

Description Exchange job ID into the index in an array according to its ID
Category DPS function
Syntax int JobIndex(int job_id);
Remarks JobIndex finds the index(position in job array) of one job when the job's ID is known. It returns the index.

24. OperationIndex

Description Exchange operation ID into the index in oper array according to its ID
Category DPS function

Syntax int OperationIndex(int oper_id);
Remarks OperationIndex finds the index(position in oper array) of one operation when the operation's ID is known. It returns the index.

25. OperationTypeIndex

Description Exchange operation type ID into the index in optype array according to its ID
Category DPS function
Syntax int OperationTypeIndex(void);
Remarks OperationTypeIndex finds the index(position in optype array) of one operation type when the operation type's ID is known. Returns the index.

26. ToDpsJWk

Description Output the detailed production schedule to DPS_J_WK table
Category DPS function
Syntax void ToDpsJWk(void);
Remarks ToDpsJWk output the values stored in structures to DPS_J_Wk table. The values include wk_job_id, optype, id, job_id in oper structure.

27. WorkerIndex

Description Exchange worker ID into the index in worker array according to its ID
Category DPS function
Syntax int WorkerIndex(int worker_id);
Remarks WorkerIndex finds the index(position in worker array) of one worker when the worker's ID is known. It returns the index.

28. WorkerRank

Description Sorts workers according to their efficiency
Category DPS function
Syntax void WorkerRank(void);
Remarks WorkerRank sorts the workers stored in worker structure according to op_eff on each operation type, as a result, each operation type has a sorted worker list.

29. WorkerAssignJob

Description Assigns workers to each job
Category DPS function
Syntax void WorkerAssignJob(void);
Remarks WorkerAssignJob assigns workers stored in wk_id to

wk_job_id in oper structure, the assignment rule is to assign the most efficient worker to the most urgent job, the job's urgency is changed all the time.

30 WorkerAssignOp

Description Assigns workers to each workstation
Category DPS function
Syntax void WorkerAssignOp(void);
Remarks WorkerAssignOp assigns workers stored in wk_id in optype structure to wk_id in oper structure, the assignment rule is to assign the most efficient worker to the most urgent workstation, the workstation's urgency is changed all the time.

4.2.3 Material flow simulation functions (31-43)

31. JobTree

Description Shows the logical layout of one job
Category Simulation function
Syntax int JobTree(int job_id, int x, int y, int h, int color);
Remarks JobTree displays the job that the ID equals to job_id in a rectangle area, the coordinate of upper corner of the area is x, y, the width and length is h. When color is 0, redraw the tree with the same color of background, this means fading the tree off the screen, otherwise display a tree structure. The function returns the width of current shown tree. This function is the subfunction of ShowTree.

32. OneWorker

Description Shows one worker
Category Simulation function
Syntax int OneWorker(int x, int y, int w, int h, int color);
Remarks OneWorker displays one worker in a rectangle area, the coordinate of upper corner of the area is x, y, the width is w and length is h. When color is 0, redraw the worker with the same color of background, this means fading the worker off the screen, otherwise displays the worker with color that indicates the efficient level of color.

33. ShowOperation

Description Displays the physical layout of the plant
Category Simulation function

Syntax void ShowOperation();
Remarks ShowOperation displays the raw material store level, all the workstations in their physical location, and the end product inventory level.

34. ShowOneOperation

Description Displays one workstation
Category Simulation function
Syntax void ShowOneOperation(int x1, int y1, int w, int oper_index, int color);
Remarks ShowOneOperation displays the workstation that the index in oper structure is oper_index, in the rectangle area with the upper corner coordinate x1, y1, width w, height 45 pixel. When color is 0, draws the workstation with background color, otherwise, show the workstation. This function is a sub-function of ShowOperation.

35. ShowTree

Description Displays the logical layout of one job with a certain scale from the original display.
Category Simulation function
Syntax void ShowTree(int w, float scale, int job_id, int color);
Remarks ShowTree re-displays the logical layout of job that has the ID value equals to job_id, original width w, new width w*scale, when color equals 0, draws the tree with the background color, otherwise displays the new job tree.

36. SimMenu

Description Shows the menu for simulation module
Category Simulation function
Syntax void SimMenu(short menu_num, char **menu);
Remarks SimMenu displays a menu which has menu_num number of menu items, the text contents of menu items are stored in two dimension array menu. No return value.

37. Simulation

Description Simulates the dynamic change for working in process inventory
Category Simulation function
Syntax void Simulation(short wkstation, short w, shor color, short j_id)
Remarks Simulation displays the working in process inventory in the end of each hour for a period a

current week, when the wkstation parameter is 0, the simulation process works on the physical layout of the plant, otherwise it does on the logical layout of a particular job with the job ID equals to j_id, w parameter indicates the width of the rectangle that the logical layout was shown in. When color is 0, redraw the working in process inventory with the same color of background, this means erasing the inventory level, otherwise display the inventory level. No return value.

38. TextOper

Description Shows the detailed information about one workstation
Category Simulation function
Syntax void TextOper(short oper_id);
Remarks TextOper displays the detailed information about workstation that the ID equals to oper_id, the detailed information include the id, operation type, the number of jobs assigned to this station, the inventory level of each job, the throughput of each job in the end of current hour, number of workers and their IDs, and their efficency.

39. WeekTask

Description Shows the tasks of jobs in current week
Category Simulation function
Syntax void WeekTask(short week);
Remarks WeekTask displays jobs' tasks planned to be finished in current week which is the output of Master Production Schedule. The format of the display is a two dimension graph, the X axis is jobs, the Y axis is tasks of each job in bundle unit. No return value.

40. WhichWorker

Description Identifies the worker who is picked by user
Category Simulation function
Syntax short WhichWorker(short m. short n);
Remarks WhichWorker finds the ID for the worker who is displayed in the rectangle that the coordinate of the upper corner of the rectangle is (m, n), returns the worker's ID.

41. WorkerPost

Description Shows the worker assignment in current week
Category Simulation function

Syntax void WorkerPost(short week);
Remarks WorkerPost displays workers' assignment in operation type that is the output of Master Production Schedule. No return value.

42. ZoomOper

Description Zooms one workstation
Category Simulation function
Syntax void ZoomOper(short id, short scale);
Remarks ZoomOper is used when the user wants to make the workstation larger. Parameter scale is the ? . No return value.

4 4 Skill prediction functions (43)

43. NonLinearFun

Description Predict the worker's skill level
Category Skill function
Syntax double NonLinearFun(double a, double b, double c, double x);
Remarks NonLinearFun returns a double value which is calculated by this formulation:
a × (1 - 1/ (1 + c × pow(x, b))).

4.3 Input and Output

This section specifies the sources and types of input and output information to be used in this system. The input data specification includes the sources, type, size, unit of measure, range of measure. The input data are specified as below.

Input data Interfaces	Type	Size	Units of measure	
Order_id	Short Number	5	-	Inter07
Customer_id	Short Number	5	-	Inter01
Order_date	Date	-	-	Inter01
Job_id	Short Number	5	-	Inter01
Style_id	Short Number	5	-	Inter01
Due_date	Date	-	-	Inter01
Penalty	Number	4	$	Inter01
Color_id	Short Number	5	-	Inter01
Size_id	Short Number	5	-	Inter01
Quantity	Short Number	6	bundle	Inter01
Finished_qty	Short Number	6	bundle	Inter01
Dozen_num	Short Number	3	dozen	Inter01
Unit_price	Number	4	$	Inter01
Cost	Number	4	$	Inter01
Subcontract	Short Number	6	bundle	Inter01

C._sub._qty	Short Number	6	bundle	Inter01
Operation_id	Short Number	5	-	Inter01
Optype_id	Short Number	5	-	Inter01
Worker_id	Short Number	5	-	Inter01
Name(worker)	Alphanumeric	10	-	Inter01
Hire_date	Date	-	day	Inter01
Overtime1-10	Short Number	3	week	Inter01
Absence1-5	Short Number	3	week	Inter01
Ini_train_days	Short Number	5	day	Inter01
Appl._levle	Binary	-	-	Inter01
Curr_week	Short Number	3	week	Inter01
Days_in_week	Short Number	3	day	Inter01
Hours_in_day	Short Number	3	hour	Inter01
Name(op_type)	Alphanumeric	10	-	Inter01
F1-F17	Number	4	-	Inter01
A, B, C	Number	4	-	Inter01
Unit_cost	Number	4	$	Inter01
Standard_time	Number	4	second	Inter01
Root_op._id	Short Number	5	-	Inter01
Prev_op_id1-3	Short Number	5	-	Inter01
Next_op_id	Short Number	5	-	Inter01
j_id	Integer	5	-	Inter07

For output, the specification includes the modules (or programs) which generates the results, destinations, quantitative terms, units of measure, layouts of examples, and graphic display reports, etc. The output data are specified as below:

Output	Type	Size	Units of measure	Generating Module
Status	Short Number	1	-	MPS & Capacity
Acceptability	Binary	-	-	MPS & Capacity
Com._subc._qty	Short Number	5	bundle	MPS & Capacity
Overtime1-10	Short Number	3	-	Capacity
Init_efficency	Number	4	bundle/sec	MPS
Curr_efficiency	Number	4	bundle/sec	MPS
This_train_status	Binary	-	-	MPS
Day_id	Short Number	5	-	MPS
Wk_id	Short Number	5	-	MPS
Optype_id	Short Number	5	-	MPS
Ws_id	Short Number	5	-	DPS or Simulation
Job_id	Short Number	5	-	DPS or Simulation
Wip_in	Short Number	5	bundle	DPS or Simulation
Wip_out	Short Number	5	bundle	DPS or Simulation
Wip_inventory	Short Number	5	bundle	DPS or Simulation
Throughput	Short Number	5	bundle	DPS or Simulation

The output display formats are listed as below. They are assigned with a unique identifier. Details are shown in display printouts.

Output Display Format	Identifier
Apparel Production Planning and Scheduling System Main Menu	Menu01
Order and Job	Order02
Characteristic of One Worker	Resource03
Workstation Information	Resource04
The Standard Time of Each Operation in One Style	Knowledge05
Operation Type and Operations	Knowledge06
Characteristic of One Style of Job	Knowledge07
Miscellaneous Information	Knowledge08
Master Production Schedule in Table	Schedule09
Master Production Schedule Weekly Production Quantity of Jobs	Schedule10
Master Production Schedule in 3D Graph	Schedule11
Master Production Schedule with Worker Assignment	Schedule12
Detailed Production Schedule of Worker Assignment	Schedule13
Detailed Production Schedule of WIP in Table	Schedule14
Detailed Production Schedule of Throughput	Schedule15
Detailed Production Schedule WIP in 2D Graph	Schedule16
Plant Physical Layout	Simulation17
Job Logical Layout	Simulation18
Skill Predictor	Skill19

4.4 Data Characteristics

This section specifies the data requirements which affect the design of the software system. The specification includes not only the input-output data introduced in the previous section, but also some major data types temporarily used by each module. It includes the information of specific data item by: the name, relative modules, description (some characters about the meaning of this data, the reason why it is introduced, the unit of measure etc.), and reference (Refer to section 4.2 for the computer program function relative to the data). The data are described alphabetically in this section. Finally, an estimate of the initial and expected growth of the storage requirements for the data and computer programs is given at the end of this section.

A

Relative Modules: Database, Skill
Description: A is an attribute of operation type. It functions as a coefficient in the skill model.

Reference: NonLinearFun

Absence1-5

Relative Modules: Database, MPS, Capacity

Description: Absence1, Absence2, Absence3, Absence4, Absence5 are attributes of a worker. One worker may be absent from work due to vocation, illness or other reason. The available capacity level of the plant is one of the major information to do master production planning and capacity planning. The workers' attendance situation will affect the capacity level of the plant. The software system maintains maximum five weeks' information about worker absence in the current schedule period.

Reference: GetAllWorker, Mps, IsOverTime

Acceptability

Relative Modules: Database, MPS

Description: Acceptability is an attribute of a job. After new orders arrived with a list of jobs, the software system helps to estimate the manufacturing lead time, whether jobs can be finished before the due date, the cost, and suggests whether the jobs are acceptable. After the user decides the acceptability of jobs either taking the suggestion of the system or according to his own experience, the system will generate a master production plan for next three months and detailed production plan for current week. When the value of Acceptability of one job is 1, it indicates this job is accepted, 0 indicates the job can not be accepted.

Reference: GetJob, Mps

Application_level

Relative Modules: Database, MPS, Capacity

Description: Application_level is one of the information kept under MISCELLANEOUS category in the database. It has two levels: test level when Application_level is 1, real time application level when its value is 0. In the test level, after every call to Master Production Schedule module, the pointer to current week is increased by one, the workers' training days on a certain operation type is increased by the working days of one week(usually five), and jobs' finished quantity is increased, if the finished quantity of one job is equal to

	the quantity the order required, the job finishes at this week; In the real time application level, all the above actions take place only when the pointer to current week is later than the system timer.
Reference:	Mps

B

Relative Modules:	Database, Skill
Description:	B is an attribute of operation type. It functions as a coefficient in the skill model. Refer to A.
Reference:	NonLinearFun

C

Relative Modules:	Database, Skill
Description:	C is an attribute of an operation type. It functions as a coefficient in the skill model. Refer to A and B.
Reference:	NonLinearFun

Color_id

Relative Modules:	Database
Description:	Color_id is an attribute of job. All the possible product color is assigned a color ID. Two items in an order are considered to be two different jobs when they are at least different in one of these aspects: Order_id, Style_id, Due_date, Color_id, Size_id.
Reference:	None

Committed_subcontract_qty

Relative Modules:	Database, Capacity
Description:	Committed_subcontract_qty is an attribute of job. The units measure is bundles. When the system finds that the current capacity of the plant can not meet the orders' requirement, Capacity module is called. Capacity module considers three alternatives of approaches to increase the capacity. One of the approaches is subcontract. The maximum allowed subcontract quantity of one job is stored in Subcontract. The actual subcontract quantity is decided by Capacity module, and stored in Committed_subcontract_qty.
Reference:	Mps

Curr_efficiency

Relative Modules:	Database, MPS, DPS, Capacity, Skill
Description:	Curr_efficiency is an attribute of workers. It indicates the worker's skill level on a

certain operation type. The units measure is bundles per minute. One of the assumptions of this system is that the worker's skill level will increase after he doing one operation type for a certain period of time(the training days on the operation type), and the skill level can be predicted using a regression model. Please refer to the Final Report, section 2.3 for detail.

Reference: GetAllWorker, SelectWorker, UpdateCurrTrain, UpdateThisTrain, InputData, WorkerRank, WorkerAssignOp, NonLinearFun

---

Curr_week

Relative Modules: Database, MPS, DPS, Simulation, Capacity

Description: Curr_week is system information kept under MISCELLANEOUS category in the database. It must be initiated periodically, for example, it is initiated by 0 in the beginning of each year. When the system timer passes one week, the value of Curr_week will be increase by 1 by Master Production Schedule module. The schedule period of Master Production Schedule module begins with the week indicated by Curr_week, The Detailed Production Schedule generates the detailed production plan for the working days of current week, and Simulation module simulates the outcomes of the schedules for the working days of current week.

Reference: GetCurrWeek, IsOverTime, Mps, SelectWorker, Umps, UpdateCurrTrain, UpdateUmsWk, UpdateUmpsDB, InputData, WorkerPost, WeekTask, ToDpsWk

---

Customer_id

Relative Modules: Database

Description: Customer_id is an attribute of order. All the customers of the plant are assigned an ID. One customer may put several orders, but one order only has one customer.

Reference: None

Day_id

Relative Modules: Database, DPS, Simulation

Description: Day_id is needed for representing a detailed production schedule. The detailed production schedule includes two aspects: one is schedule for worker assignment to job and to workstation, another is for keeping track of working in process inventory. Two tables are

	designed to record these detailed information respectively: DPS_J_WK and DPS_WIP. If the number of working days in one week is five, the value of Day_id is from one to five.
Reference:	ToDpsWk, UpdateWip

Days_in_week

Relative Modules:	Database, MPS, DPS, Simulation, Capacity
Description:	Days_in_week is system information kept under MISCELLANEOUS category in the database. It indicates the number of working days in a week adopted by the plant. This information affects the available capacity of the plant, and the growth of workers' skill level, so that it influences the master production schedule, detailed production schedule, capacity plan, and the simulation solution.
Reference:	Umps, ToDpsWk, UpdateCurrTrain, Simulation

Dozen_num

Relative Modules:	Database, MPS, DPS, Simulation
Description:	Dozen_num is an attribute of job. One job in an order must has quantity stored in Quantity field, the measures of quantity is bundles. One bundle of different job may has different dozens of product, depending on the size of the product. The number of dozens in one bundle of a job is stored in Dozen_num.
Reference:	GetJob, InputData

Due_date

Relative Modules:	Database, MPS, Capacity
Description:	Due_date is an attribute of job. Jobs in the same order may have different due date. The due date of one job is given by user. It is assumed that changing the due date of job is the most expensive way to deliver the job on time compared to overtime, subcontract, and new hire. MPS module generates the master production schedule under the discipline of finishing the job with the earliest due date. The value of Due_date is represented by DATE type supported by Paradox, for example, Jan.1,1995 is represented by 1/1/1995. Another field of job named Due_week indicates the corresponding week ID of due date when the first week ID of the year is assumed to be 1.
Reference:	GetJob

F1-F17

Relative Modules:	Database, MPS, DPS, Simulation, Skill

Description:	F1 to F17 are attributes of operation type. They are used as coefficients by Skill model to predict the growing of workers' skill level on the operation type. Please refer to Final Report section 2.3 for detail.
Reference:	GetAllOptype, InputData, LearnmLinear

Finished_qty

Relative Modules:	Database, MPS
Description:	Finished_qty is an attribute of job. A job in an order has an attribute Quantity which stores the required amount of the final product of this job. At the end of each week, some jobs have be finished a certain amount of quantity that is accumulated into the field Finished_qty of the corresponding jobs. When the values of Quantity and Finished_qty are equal for a certain job at the end of a week, the job is finished, and the Status of this job must be set to 1, this means the job is existing job but finished, it need not be considered in the next MPS period.
Reference:	FinishedQty, RecovereFiniQty, UpdateFiniQty

Hire_date

Relative Modules:	Database
Description:	Hire_date is an attribute of worker. Hire_date records the date when the worker was hired. The worker's initial efficiency and initial training days on each operation type are evaluated at this date.
Reference:	None

Hours_in_day

Relative Modules:	Database, MPS, DPS, Simulation, Capacity
Description:	Hours_in_week is system information kept under MISCELLANEOUS category in the database. It indicates the number of working hours in a working day adopted by the plant. This information affects the available capacity of the plant, so that it influences the master production schedule, detailed production schedule, capacity plan, and the simulation solution.
Reference:	Umps, Simulation

Ini_efficency

Relative Modules:	Database, MPS
Description:	Ini_efficency is a skill level measurement for one worker doing a certain operation type at the time he was hired. The units measure is

bundles per minute. Initial efficiency is tightly related to initial training days of the same worker on the same operation type. At the statistic point of view, one worker's skill efficiency on one operation type will increase as the training days grows, the growth of efficiency has a certain pattern which depends on the nature of the operation type.

Reference: GetAllWorker

Ini_train_days

Relative Modules: Database, MPS

Description: Ini_train_days stores the number of training days one worker has been done on a certain operation type at the hire date. As introduced in Ini_efficency, these two variables are tightly related in the statistic point of view, usually, the initial efficiency of one worker doing one operation type can be estimated from the training days he has been done on this operation type.

Reference: GetAllWorker

J_id

Relative Modules: Simulation

Description: J_id is a variable needed to acquire a job ID from user so as to show the job's logical operation structure in the Simulation module In the Simulation module, a menu is developed. When the user pick TREE menu to do some operations on the job tree, a dialogue box will be pop out in the top of the screen, it asks the user to input one job ID. After user type an ID, the value is directly accepted to variable J_id.

Reference: JobTree, ShowTree, Simulation

Job_id

Relative Modules: Database, MPS, DPS, Simulation, Capacity

Description: Job_id is an attribute of job, it also needed to represent the schedules including master production schedule and detailed production schedule. One item in an order is treated as an unique job when it is at least different in one of these aspects from another item: Order_id,. Style_id, Due_date, Color_id, Size_id, and the unique job is assigned an unique job ID stored in Job_id.

Reference: GetJob, RecoverFiniQty, UpdateFiniQty, UpdateUmpsDB, InputData, JobIndex, ToDpsJWk,

JobTree, ShowTree, Simulation

Name(operation type)

Relative Modules:	Database
Description:	Name(for operation type) is an attribute of operation type. Every operation type has its name.
Reference:	None

Name(worker)

Relative Modules:	Database
Description:	Name(for worker) is an attribute of worker. Every worker has his/her name.
Reference:	None

Next_op_id

Relative Modules:	Database, MPS, DPS, Simulation
Description:	Next_op_id is an attribute of operations in a style. A production style is consist of a set of operations with a certain sequence pattern. One operation in a style has only one subsequent operation, the ID is indicated by Next_op_id. An operation may has different subsequent operation in different production style.
Reference:	GetStyle, InputData

Operation_id

Relative Modules:	Database, MPS, DPS, Simulation
Description:	Operation_id is an attribute of operation. Every operation the plant can do is assigned an ID. The operations are dependent with each other to make a product, so that the Operation_id is also used to represent production styles, the schedules including MPS and DPS.
Reference:	GetStyle, GetOperId, InputData, OperationIndex, ToDpsWk, WorkerAssignOp, ShowOneOperation, ShowOperation, TextOper, ZoomOper

Optype_id

Relative Modules:	Database, MPS, DPS, Simulation
Description:	Optype_id is an attribute of operation type. Each operation type which the plant can do is assigned an ID. It also is an attribute of operation, each operation has its operation type. The master production schedule includes the information about the workers's assignment to operation type.

Reference: GetAllOprype, Umps, UpdateUmpsWk

Order_date

Relative Modules: Database, MPS

Description: Order_date is an attribute of order. In this date, the jobs in the order are new jobs, as a result, the Status of these jobs are set to be 3 (1 indicates finished jobs, 2 indicates existing job but not finished job, 3 indicates new jobs). If the acceptance of new jobs will bring about that one or more of the existing jobs' may not be finished before their due date, the new jobs are recommended to be rejected.

Reference: GetJob

Order_id

Relative Modules: Database

Description: Order_id is an attribute of order. Every order ever received by the plant is assigned an unique ID and recorded in the Order table.

Reference: None

Overtime1-10

Relative Modules: Database, MPS, DPS, Capacity

Description: Overtime1 to Overtime10 are attributes of worker. The workers of the plant may have an agreement with the plant about the overtime. Overtime1 to Overtime10 are ten fields to record the week IDs in which the worker is agree to overtime. The available capacity level of the plant is one of the major information to do master production planning and capacity planning. The workers' overtime situation will affect the capacity level of the plant. The software system maintains maximum ten weeks' information about worker overtime in the current schedule period.

Reference: Mps

Penalty

Relative Modules: Database, MPS

Description: Penalty is an attribute of job, it stores the dollar penalty per day that will occur if the job can not be finished before the due date. This penalty contributes the cost to finish the job.

Reference: Mps

Prev_op_id1-3

Relative Modules: Database, MPS, DPS, Simulation

Description:	Next_op_id1, Prev_op_id2, Prev_op_id3 are attributes of operations in a style. A production style is consist of a set of operations with a certain sequence pattern. The software system assumes that one operation in a style has at most three previous operations, the IDs are indicated by Prev_id1, Prev_id2, Prev_id3. One operation may has different set of previous operations in different production style.
Reference:	GetStyle, InputData

Quantity

Relative Modules:	Database, MPS
Description:	Quantity is an attribute of job. One job coming with an order must has a certain amount of products that required to be done. The amount is stored in Quantity with units measure bundles. The products consists of this quantity are same in following aspects: order ID, style ID, due date, penalty, color ID, size ID, dozen number, unit price, status, cost, subcontract.
Reference:	GetJob

Root_operation_id

Relative Modules:	Database, MPS, DPS, Simulation
Description:	Root_operation_id is an attribute of style. It records the final assembly operation ID of one production style. One style ID must has only one root operation ID, but one root operation ID may be the different style's root ID. All the operations of one style can be found in the STYLE_OP table if we know the root operation ID.
Reference:	GetStyle, InputData

Size_id

Relative Modules:	Database, MPS
Description:	Size_id is an attribute of job. All the possible product size is assigned an size ID. Two items in an order are considered to be different jobs when the size ID is different, even if all of the other attributes are the same. Some operations in the same style with different product size may have different processing time, operation to do bigger product usually need longer standard processing time.
Reference:	GetJob, InputData

Standard_time

Relative Modules:	Database, MPS, DPS, Simulation
Description:	Standard_time represents the processing time needed by an standard worker to finish an operation of a particular style and size. The standard processing time of one operation may different if the style or the size of the product is different. The information of standard processing time is kept in STIME table in Database, and is needed to estimate the required capacity of jobs. Master production schedule is developed by comparing the required capacity of jobs and the available capacity of the plant. Detailed production schedule is generated by assigning the most efficient worker to the most urgent operation(longest aggregated processing time).
Reference:	GetStyle, SelectStime, InputData, WorkerAssignJob, WorkerAssignOp

Status

Relative Modules:	Database, MPS
Description:	Status is an attribute of job. It has three value: 1, indicates the job has already been finished, it usually set by MPS module when the module finds that the value of the job's Quantity and Finished_qty are equal to each other; 2, indicates the job is existing but needs a few more weeks to be finished; 3, indicates the job is new. When the value of Status is 1 or 2, the corresponding value of Acceptability must be 1.
Reference:	GetJob, Mps

Style_id

Relative Modules:	Database, MPS, DPS, Simulation
Description:	Style_id is an attribute of product style. All the styles the plant deals with are assigned an ID, and the style structures are captured by two dimension tables. One style has only one root operation(final assembly). Knowing the style ID and its root operation ID, one can find all the operations of the style through STYLE_OP table.
Reference:	GetStyle, InputData

Subcontract

Relative Modules:	Database, Capacity
Description:	Subcontract is an attribute of job. It indicates the maximum amount of the job that can be transferred to other plants as

subcontract, the value is required from user. The unit measure is bundle. Please refer to Committed_subcontract_qty.

Reference: Mps

This_train_status

Relative Modules: Database, MPS

Description: This_train_status tells if one worker is doing one operation type in this week. This information is needed for keeping track of one worker's training days on one operation type. If one worker's corresponding This_train_status is 1 on a certain operation type, indicates that the worker is assigned to the operation type. His/her Curr_train_days will be increased by a certain number of days(Days_in_week in MISCELL table), otherwise he/her is not doing the operation type.

Reference: UpdateCurrTrain, UpdateThisTrain

Throughput

Relative Modules: Database, MPS, DPS, Simulation

Description: Throughput is introduced to represents the master production schedule. It tells how many bundles of each job are scheduled to be finished in coming weeks in the schedule period.

Reference: Umps, UpdateUmpsDB, IniJobTask

Unit_cost

Relative Modules: Database, MPS

Description: Unit_cost is an attribute of job. It indicates the manufacturing cost to make one product of the job if the job is finished on time. Unit measure is U.S. dollar.

Reference: Mps

Unit_price

Relative Modules: Database

Description: Unit_price is an attribute of job. It indicates the price of one product of this job sold to the costumer. It is given by user.

Reference: None

Wip_in

Relative Modules: Database, DPS, Simulation

Description: Wip_in is an attribute of workstation, and is introduced for representing a detail production schedule. It records the aggregate input of one job in one workstation in a particular day. The unit

measure is bundle.
Reference: UpdateWip, Simulation

Wip_inventory
Relative Modules: Database, DPS, Simulation
Description: Wip_inventory represents the inventory level of one job in one workstation at the end of one day. The unit measure is bundle. Please refer to Wip_in.
Reference: UpdateWip, Simulation

Wip_out
Relative Modules: Database, DPS, Simulation
Description: Wip_out records the aggregate output of one job in one workstation in a certain day. The unit measure is bundle. For one job in one workstation, the inventory at the beginning of the day plus the input in the whole day minus the output of the day must be equal to the inventory remaining for the next day.
Reference: UpdateWip, Simulation

Worker_id

Relative Modules: Database, MPS, DPS, Capacity
Description: Worker_id is an attribute of worker. Every worker in the plant is assigned an unique ID.
Reference: GetAllWorker, SelectWorker, Umps, UpdateCurrTrain, UpdateThisTrain, UpdateUmpsWk, Inputdate, WorkerIndex, WorkerRank, WorkerAssignJob, WorkerAssignOp, WorkerPost, WhichWorker

4.5 Failure Contingencies

This section provides a discussion of the requirements for recovering the software system from a failure. Before installing the system, a back up should be made in case of hardware or software crash. The back up should includes files with following extensions:

- BGI (provided by Turbo C, and needed to show graphics)
- DB (Paradox data files)
- EXE (system executable files)
- FSL (Paradox forms)
- PX (index files)
- SSL (Paradox scripts)
- XG0 (secondary key)
- YG0 (secondary key)

All the above files except the DB files can not be edited. The DB files are data files. They include resources, knowledge, schedules and other information; thus they are always updatable. If the system crashes because of some hardware or an unknown problem

such that the system cannot behave normally, the system should be installed again. If the computer is shut down during a master production scheduling session, the DB files are be updated incompletely and as a result, some data may be inconsistent. The most recent DB backup files are then needed to re-create the working directory.

4.6 Design Requirements

The system is designed with the object-oriented and open architecture concepts so that the system can be easily maintained and expanded. As shown the Figure 1 in Chapter 2, the system consists of three layers: the GUI, system application components, and the database. Each of these layers is functionally separately and designed independently. The database layer is designed to store all information that may be used to support system functionalities. Based on the need for support of individual application modules, all data are decomposed into pieces and re-organized into tables to avoid redundancy and promote access efficiency.

All system components are designed to reflect system functionalities. The components can be classified into two types: shared components which may be called by and support other components. These modules include skill prediction, capacity planning, and training scheduling, etc. The other type of components are application components which are attached to the GUI to provide system functionalities. These components include the material flow simulation module, and the MPS module, etc. A shared component could be also an application component. The GUI layer provides a friendly user interface, through which all system functionalities can be easily operated. Output data can also be shown on the screen in graphics or texts.

4.7 Computer Security Requirements

N/A

4.8 Human Performance Requirements

N/A

5. TEST AND QUALIFICATION REQUIREMENTS

This chapter summarizes the tests of the system in section 5.1 and the requirments for testing the system in section 5.2.

5.1 Introduction

Both unit tests and integration tests were conducted. All tests were carried out on a Dell 486DX2 under MS/DOS 6.2 and MS/Windows 3.1. Test data were collected from Bernard Cap and

other apparel firms. They were manually converted into Paradox tables. Unit tests including the skill prediction, MPS, DPS, and simulation, were conducted according to these tables. The integration test was carried out under Paradox environment. The following sections will further detail the tests.

5.1.1 Unit tests

Unit tests were conducted with system components of skill prediction, MPS, DPS, and simulation. This section describes each unit test with a focus on its input, output, and testing procedure. The skill prediction module was applied to all operation types developed in this study. The input to the module is operation type. The output of the module is a curve which reflects skill improvement of an average worker along with the number of work days at this operation type. The curve shows the skill is improved sharply at beginning period and then level off at the end. It matches the regression model developed in this study.

The input data to the MPS module for the test were: 1) four new jobs in *JOB.DB* of two orders in *ORDER.DB*; 2) 100 workers in *WORKER.DB* (all are available and the efficiency at each operation type is known in *WK_OPTP.DB*); 3) 10 operation types in *OPTP.DB* (each is associated with seventeen coefficients); 4) two garment styles in *STYLE.DB and STYLE_OP.DB*; 5) two garment sizes in *JOB.DB*; 6) 30 workstations in *OPER.DB* (each does only one operation type); and 7) the standard processing time of each operation in *STIME.DB*. The unit test was run by invoking the MPS.EXE program under the DOS prompt, which got input data from Paradox tables and generated the schedule, and stored in tables *UMPS.DB* and *UMPS_WK.DB*, showing: 1) the weekly production quantity for each job; and 2) weekly worker assignment by operation type. These output data were manually examined.

The unit test of the DPS module was carried out with the master production schedule as an input. The following related data were used: 1) workers in *WORKER.DB* and *UMPS_WK.DB*; 2) the operation types in *OPTP.DB*; 3) the garment styles in *STYLE.DB, and STYLE_OP.DB*; 4) the garment sizes in *JOB.DB*; 5) the workstations stored in *OPER.DB*; and 6) the standard processing times for the operations in *STIME.DB*. The test run was carried out by invoking the DPS.EXE file under the DOS prompt, which got input data from the above Paradox tables, generated a detailed production schedule, and stored in the *DPS_J_WK.DB* and *DPS_WIP.DB* table. The DPS output contains: 1) worker assignment to each workstation; 2) worker assignment to each job; and 3) daily WIP inventory at each workstation. These output data were manually verified.

The unit test of the simulation module was conducted with the output data from the MPS and DPS modules. The purpose of the simulation module is to validate the static production plan under

a dynamic environment. The testing was carried with the following input data: 1) worker to workstation assignment from *DPS_J_WK.DB*, *WORKER.DB*, and *OPER.DB*); 2) worker to job assignment from *DPS_J_WK.DB*, *WORKER.DB*, and *JOB.DB*); 3) operation types from *OPTP.DB*; 4) garment styles from *STYLE.DB*, and *STYLE_OP.DB*; 5) garment sizes in *JOB.DB*; and 6) standard processing time from *STIME.DB*. The test was carried out by invoking the SIMULATION.EXE program under the DOS prompt, which got input data from the above Paradox tables and showed the layouts and dynamic simulation results on the monitor. All the module features were manually verified.

5.1.2 Integration test

The integration test was done in the Paradox/Windows environment. The database and each application module including MPS, DPS, simulation, and skill were integrated through the graphic user interface. All data were loaded into the database as tables. The test was carried out in the following sequence: 1) Run the skill module by selecting the *SewingSkillPredictor* button from the *SystemModules* menu to check if it works correctly through the GUI. 2) Choose the MPS button from the *SystemModules* menu to invoke MPS module. With a master production plan generated, the check was made to verify the display of 2D bar charts and 3D bar chars by selecting *MpsJob2D*, *MpsJob3D*, and *MpsJobTable* buttons from the *Schedule* menu. 3) Run the DPS module by clicking the DPS button from the GUI. The result was then checked by opening corresponding tables by choosing the *DpsWorker* and *DpsWip* items from the GUI. Manual verification was also conducted to check the result of the generated worker assignment and WIP inventories. 4) Run the simulation module by selecting the *ShopFlowSimulator* button from the *SystemModules*. The results were also manually inspected and compared to previous simulation results.

5.2 Test Requirements

To conduct a test, the following forms should be prepared. Each of these forms can be opened by clicking a corresponding form name in the GUI. The data can then be manually typed in.

1. Order
2. Worker
3. Wkstation
4. Stime
5. Style
6. OperType
7. Miscell

A set of test data for the above data types is provided in Tables 1-7. Once the data in all above tables are stored in the database, the master production schedule and detailed production schedule, for example, can be activated to generate production schedules.

Table 8 shows the generated master production schedule which specifies the scheduled production quantity for each job in the following weeks. In this example, four jobs are scheduled according to their urgency (due date) and order quantity. This schedule shows that the weekly production quantity for each job is relative stable. It implies that workers could remain at the same workstation and improve their skill efficiency by performing the same operation. It also promotes stability and reduce manager's burden of constantly re-allocating workers. The detailed production schedule shows worker's assignment to operation and job and the WIP inventory at each workstation. It is consistent with the master production schedule but creates a more detailed worker to workstation and job assignment list.

Table 1 : Order

Order_id	4001		4002	
Job_id	1	2	3	4
Style_id	87	89	87	89
Due_date	2/1/95	4/1/95	4/23/95	5/1/95
Unit_penalty	$35	$40	$40	$35
Color_id	4	2	1	1
Size_id	2	1	1	2
Quantity	8,000	9,000	5,500	7,000
Finished_qty	0	0	0	0
Dozen_number/bundle	5	2	2	5
Unit_price	$40	$60	$60	$40
Status	2	2	2	2
Subcontract	0	0	0	0

Table 2: Operation type-Operation

No.	Operation type_id	Operation_id
1	1	12, 20, 24
2	2	2, 13
3	3	11, 14, 23
4	4	3, 10, 19
5	5	8, 15, 16, 21, 27
6	6	18, 28
7	7	4, 6, 29
8	8	5, 7, 22
9	9	9, 25, 30
10	10	1, 17, 26

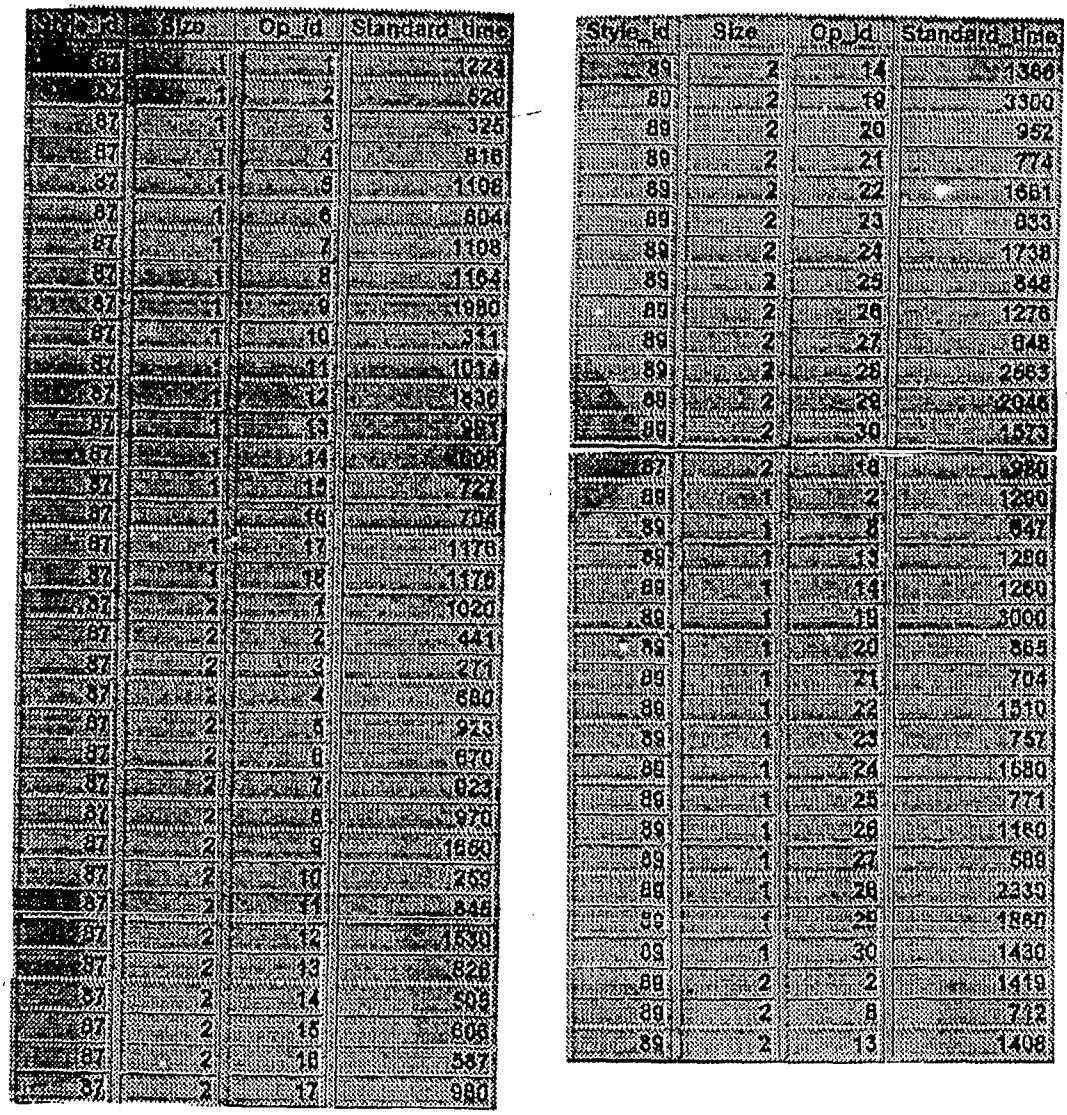
Table 3 : Sewing Workers

Worker_id	Optype_id	Ini_trainning_days	Efficiency
1	1,2,4,6	12,120,8,70	.41, .89, .47, 1.10
2	1,2,3,4	25,220,36,24	.71, .91, .95, .89
3	1,2,3,7	38,250,46,325	.86, .91, 1.0, .85
4	1,2,3,4	27,68,14,12	.74, .86, .65, .63
5	1,2,3,8	16,180,380,17	.52, .91, 1.12,.23
6	1,3,5,9	22,27,65,24	.65, .88, .81,1.23
7	2,3,6,8	160,64,28,25	.9, 1.04, .95, .34
8	1,2,4,10	24,38,12,286	.69, .78, .63, .99
9	2,3,5,9	16,220,36,260	.56,1.12, .62,1.34
10	3,4,7,10	28,6,29,34	.89, .36, .73, .81
11	2,5,8,10	24,18,36,17	.68, .38, .48, .65
12	4,6,7,9	164,362,178,14	1.19,1.20 .84,1.13
13	5,6,8,10	16,286,12,28	.34,1.19, .15, .77
14	4,7,266,8	43,52,286,8	1.04, .79,1.12, .96
15	1,6,7,10	27,28,7,220	.74, .95, .37, .98
16	3,5,6,8	176,18,9,14	1.11, .38, .61, .18
17	2,4,7,10	6,15,288,25	.24, .72, .85, .75
18	3,5,8,9	36,192,15,32	.95,1.00, .20,1.26
19	5,6,7,9	21,14,27,19	.43, .76, .71,1.19
20	4,5,6,7	112,57,23,31	1.16, .77, .90, .74
21	6,7,9,10	20,7,17,19	.86, .37,1.17, .68
22	2,4,9,10	31,7,8,41	.74, .42, .96, .84
23	1,3,5,6	318,15,25,34	1.19, .68, .49, .99
24	4,7,8,10	35,182,24,42	.99, .84, .33, .85
25	2,5,7,9	4,16,6,65	.15, .34, .33,1.31
26	3,6,8,10	9,25,22,186	.47, .92, .30, .96
27	1,4,7,9	15,16,116,56	.50, .79, .83,1.30
28	2,4,6,8	12,18,68,512	.46, .79,1.10,1.17
29	3,5,7,8	14,17,68,94	.65, .36, .81,1.32

30	4,6,8,10	17,96,327,119	.17,1.13,1.13, .95
31	1,3,5,7	57,38,33,29	.98, .96, .59, .73
32	2,4,6,8	419,74,29,17	.92,1.13, .96 .23
33	3,5,7,9	42,56,64,64	.98, .77, .81,1.33
34	4,6,8,10	116,17,42,15	1.17, .82, .54, .62
35	1,3,5,7	49,31,28,17	.94, .92, .53, .62
36	2,4,6,8	12,19,32,147	.46, .81, .98, .99
37	3,5,7,9	23,14,11,174	.93, .30, .51,1.33
38	4,6,8,10	42,36,31,15	1.04,1.00, .42, .62
39	1.3,5,7	65,23,46,57	1.01, .83, .71, .80
40	2,3,4,6,8	2,5,5,43,184	.06,.26,.3,1.03,1.04
41	3,5,7,9	1,62,135,412	.03, .80, .84,1.34
42	4,6,8,10	5,19,27,369	.31, .85, .37, .99
43	1,3,5,7	74,24,15,20	1.04, .84, .32, .66
44	2,4,6,8	39,21,15,10	.79, .84, .78, .12
45	3,5,7,9	46,32,35,96	1.00, .58, .75,1.32
46	4,6,8,10	54,26,18,30	1.08, .93, .24, .79
47	1,3,5,7	42,86,25,37	.89,1.07, .49, .76
48	2,4,6,8	36,41,35,119	.77,1.03,1.00, .93
49	3,5,7,9	20,11,7,78	.78, .24, .37,1.32
50	4,6,8,10	15,20,25,117	.72, .86, .34, .95
51	1,3,5,7	10,15,29,78	.35, .68, .55, .82
52	2,4,6,8	15,20,27,176	.54, .83, .94,1.03
53	3,5,7,9	25,74,34,65	.86, .84, .75,1.31
54	4,6,8,10	3,7,10,15	.18, .53, .12, .62
55	1,3,5,7	8,11,32,116	.28, .55, .58, .83
56	2,4,6,8	14,10,24,36	.52, .56, .91, .48
57	3,5,7,9	27,30,57,143	.88, .56, .80,1.33
58	4,6,8,10	42,52,117.256	1.04,1.06, .93, .99
59	1,3,5,7	46,33,49,86	.92, .93, .73, .82
60	2,4,6,8	25,22,43,49	.69, .86,1.03, .60
61	3,5,7,9	114,286,28,12	1.09,1.03, .72,1.09

62	4,6,8,10	18,29,12,143	.79, .96, .15, .96
63	1,3,5,7,9	52,28,43,94,56	.96,.89,.69,.83,1.3
64	2,4,6,8	24,27,74,12	.68,1.06,1.11, .15
65	3,5,7,9	18,36,184,16	.75, .62, .84,1.16
66	4,6,8,10	11,256,184,5	.59,1.19,1.04, .30
67	1,3,5,7	212,47,21,6	1.17,1.00, .43, .33
68	2,4,6,8	9,84,21,79	.37,1.14, .88, .79
69	3,5,7,9	5,29,114,234	.26, .55, .83,1.33
70	4,6,8,10	9,15,26,74	.51, .78, .36, .92
71	1,3,5,7	13,147,5,16	.44,1.10, .10, .61
72	2,4,6,8	9,12,57,46	.37, .63,1.08, .58
73	3,5,7,9	7,165,46,312	.38, .98, .78,1.34
74	4,6,8,10	8,31,38,14	.47, .97, .50, .60
75	1,3,5,7	6,3,9,54	.20, .14, .19, .80
76	2,4,6,8	7,31,46,57	.29, .96,1.05, .66
77	3,5,7,9	42,56,34,41	.98, .77, .75,1.29
78	4,6,8,10	242,4,2,76	1.20, .36, .02, .92
79	1,3,5,7	4,9,15,31	.13, .47, .32, .74
80	2,4,6,8	74,56,21,46	.87,1.09, .86, .58
81	3,5,7,9	13,12,15,49	.70, .26, .59,1.30
82	4,6,8,10	7,12,164,10	.42, .71,1.02, .55
83	1,3,5,7	5,9,116,43	.17, .47, .93, .78
84	2,4,6,8	5,7,86,25	.20, .42,1.12, .34
85	3,5,7,9	15,29,85,192	.68, .55, .82,1.33
86	4,6,8,10	8,41,31,38	.47,1.03, .42, .83
87	1,3,5,7	9,11,15,25	.31, .55, .32, .70
88	2,4,6,8	86,54,126,46	.88,1.08,1.06, .58
89	3,5,7,9	421,12,7,15	1.12, .26, .37,1.15
90	4,6,8,10	54,10,14,16	1.08, .65, .18, .63
91	1,3,5,7	15,11,8,37	.50, .55, .17, .76
92	2,4,6,8	25,11,15,54	.69, .59, .78, .64
93	3,5,7,9	45,72,18,7	.99, .84, .64, .91

94	4,5,7,9,10	40,11,19,36,65	1.02,.24,.65,1.28,.9
95	1,3,5,7	42,25,64,146	.89, .86, .81, .84
96	2,4,6,8	17,19,25,56	.58, .81, .92, .66
97	3,5,7,9	12,24,3,47	.59, .48, .17,1.29
98	4,6,8,10	10,16,25,38	.56, .80, .34, .83
99	1,3,5,7,9	17,25,53,62,174	.55,.86,.75,.81,1.33
100	2,4,6,8	53,59,49,186	.83,1.10,1.06,1.05



Style_id	Size	Op_id	Standard_time
[illegible]	1	1	1224
[illegible]	1	2	620
[illegible]	1	3	325
[illegible]	1	4	816
[illegible]	1	5	1106
[illegible]	1	6	804
[illegible]	1	7	1108
[illegible]	1	8	1164
[illegible]	1	9	1980
[illegible]	1	10	311
[illegible]	1	11	1014
[illegible]	1	12	1836
[illegible]	1	13	981
[illegible]	1	14	1606
[illegible]	1	15	721
[illegible]	1	16	704
[illegible]	1	17	1176
[illegible]	1	18	1170
[illegible]	2	1	1020
[illegible]	2	2	441
[illegible]	2	3	271
[illegible]	2	4	680
[illegible]	2	5	923
[illegible]	2	6	670
[illegible]	2	7	923
[illegible]	2	8	970
[illegible]	2	9	1660
[illegible]	2	10	759
[illegible]	2	11	846
[illegible]	2	12	1530
[illegible]	2	13	826
[illegible]	2	14	509
[illegible]	2	15	806
[illegible]	2	16	587
[illegible]	2	17	980

Style_id	Size	Op_id	Standard_time
89	2	14	1366
89	2	19	3300
89	2	20	952
89	2	21	774
89	2	22	1661
89	2	23	833
89	2	24	1738
89	2	25	846
89	2	26	1276
89	2	27	848
89	2	28	2663
89	2	29	2046
89	2	30	1573
89	2	18	980
89	1	2	1290
89	1	8	947
89	1	13	1280
89	1	14	1260
89	1	15	3000
89	1	20	865
89	1	21	704
89	1	22	1510
89	1	23	757
89	1	24	1680
89	1	25	771
89	1	26	1160
89	1	27	589
89	1	28	2339
89	1	29	1860
89	1	30	1430
89	2	2	1419
89	2	8	712
89	2	13	1408

Table 4: STIME

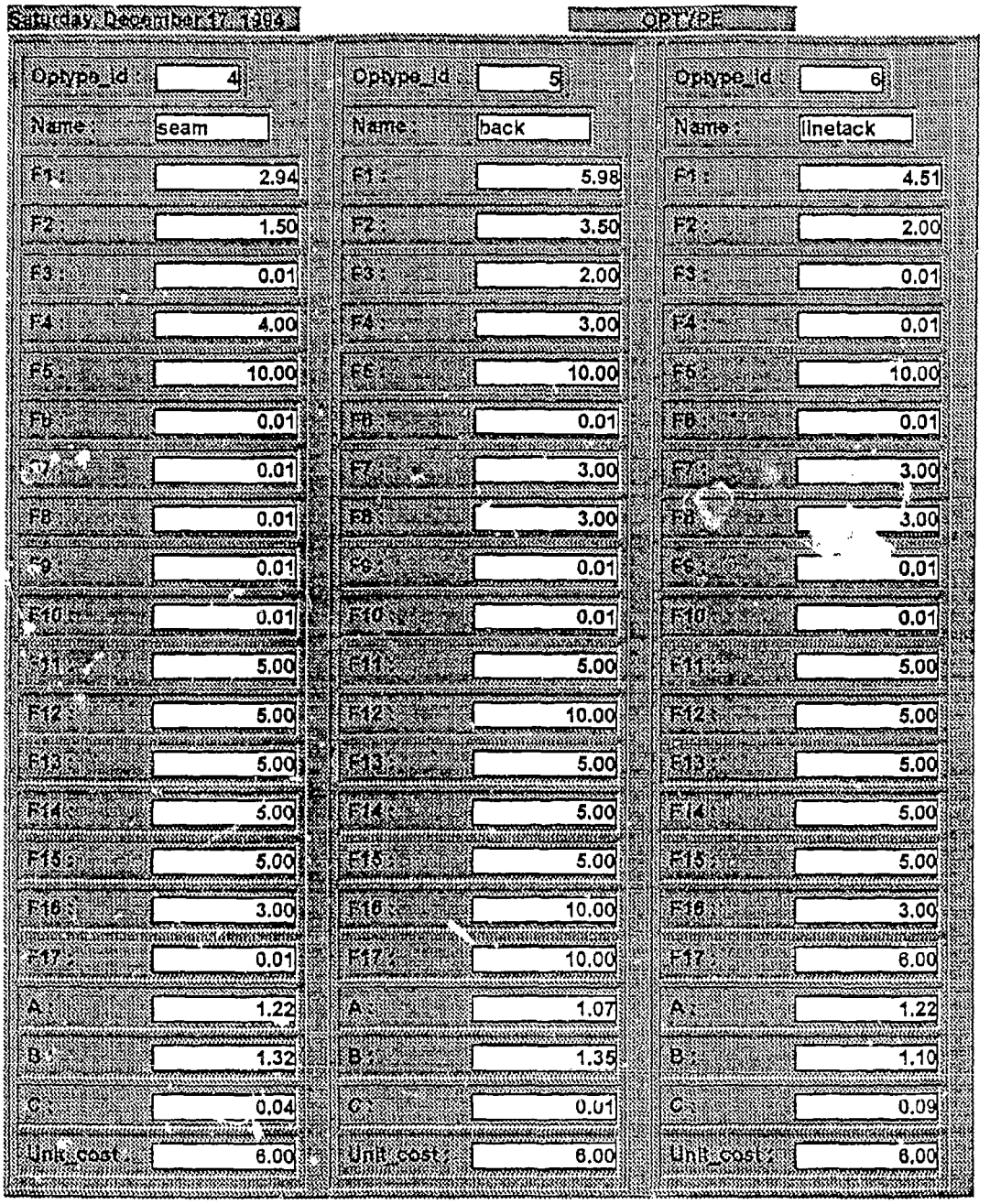
Table 5: Styles

Style_id	Root_operation_id
87	18
89	30

Saturday, December 17, 1983 Table 6.1: OPTYPE

Optype_id:	1	2	3
Name:	collar	clew	placket
F1:	2.55	4.00	6.80
F2:	1.25	1,025.00	2.00
F3:	0.01	1.00	0.01
F4:	0.50	2.00	2.50
F5:	10.00	10.00	10.00
F6:	0.01	0.01	0.01
F7:	0.01	0.01	3.00
F8:	0.01	0.01	3.00
F9:	5.00	0.01	0.01
F10:	0.01	0.01	0.01
F11:	5.00	5.00	10.00
F12:	10.00	5.00	10.00
F13:	5.00	5.00	5.00
F14:	5.00	5.00	10.00
F15:	5.00	5.00	5.00
F16:	10.00	3.00	10.00
F17:	3.00	3.00	3.00
A:	1.21	0.92	1.13
B:	1.35	1.49	1.44
C:	0.02	0.03	0.03
Unit_cost:	8.00	8.00	7.00

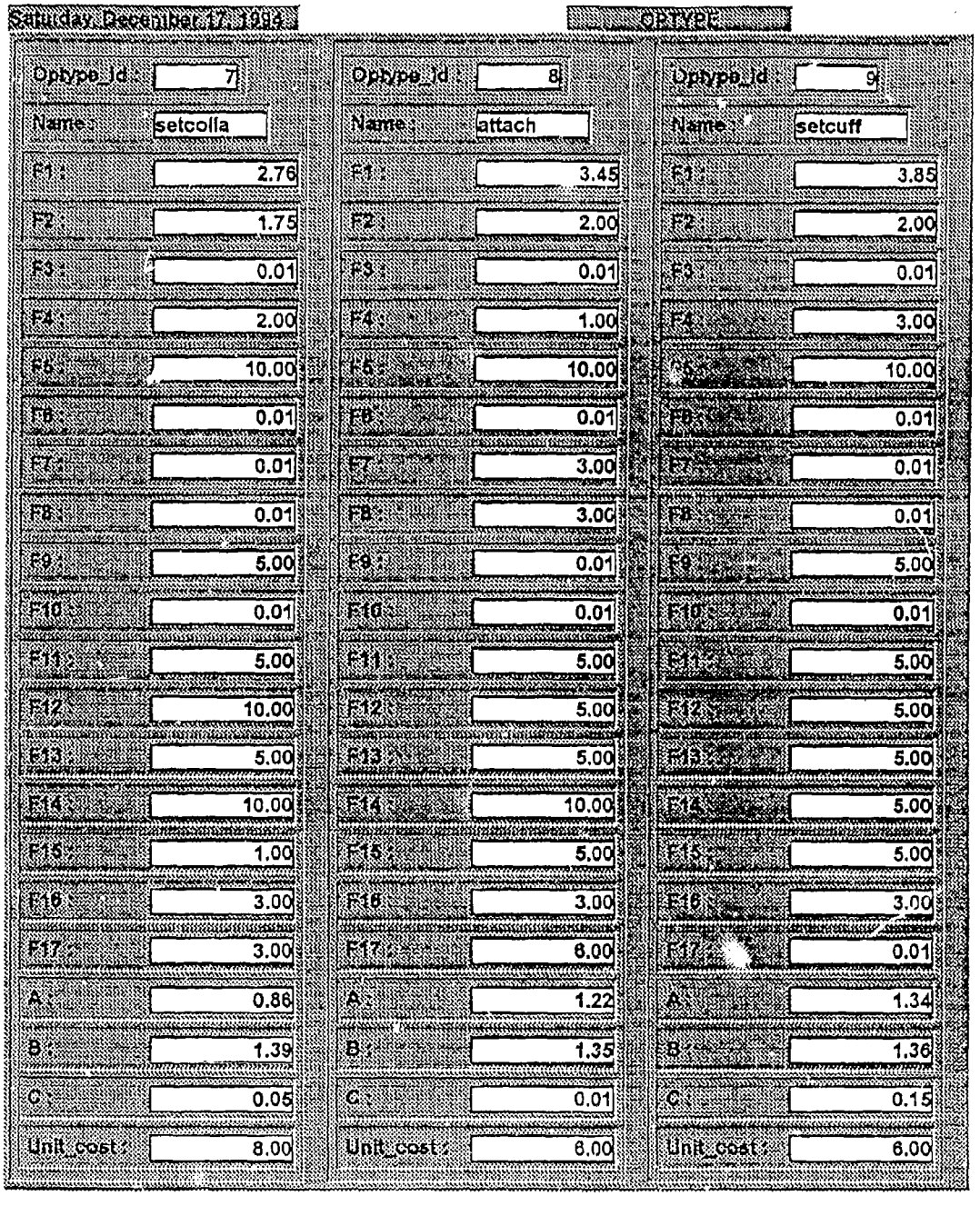
Table 6.2: OPTYPE



Saturday, December 17, 1994 — OPTYPE

	Optype_id: 4	Optype_id: 5	Optype_id: 6
	Name: seam	Name: back	Name: linetack
F1:	2.94	5.98	4.51
F2:	1.50	3.50	2.00
F3:	0.01	2.00	0.01
F4:	4.00	3.00	0.01
F5:	10.00	10.00	10.00
F6:	0.01	0.01	0.01
F7:	0.01	3.00	3.00
F8:	0.01	3.00	3.00
F9:	0.01	0.01	0.01
F10:	0.01	0.01	0.01
F11:	5.00	5.00	5.00
F12:	5.00	10.00	5.00
F13:	5.00	5.00	5.00
F14:	5.00	5.00	5.00
F15:	5.00	5.00	5.00
F16:	3.00	10.00	3.00
F17:	0.01	10.00	6.00
A:	1.22	1.07	1.22
B:	1.32	1.35	1.10
C:	0.04	0.01	0.09
Unit_cost:	6.00	6.00	6.00

Table 6.3: OPTYPE



Saturday, December 17, 1994

OPTYPE

Optype_id :	7	Optype_id :	8	Optype_id :	9
Name :	setcolla	Name :	attach	Name :	setcuff
F1 :	2.76	F1 :	3.45	F1 :	3.85
F2 :	1.75	F2 :	2.00	F2 :	2.00
F3 :	0.01	F3 :	0.01	F3 :	0.01
F4 :	2.00	F4 :	1.00	F4 :	3.00
F5 :	10.00	F5 :	10.00	F5 :	10.00
F6 :	0.01	F6 :	0.01	F6 :	0.01
F7 :	0.01	F7 :	3.00	F7 :	0.01
F8 :	0.01	F8 :	3.00	F8 :	0.01
F9 :	5.00	F9 :	0.01	F9 :	5.00
F10 :	0.01	F10 :	0.01	F10 :	0.01
F11 :	5.00	F11 :	5.00	F11 :	5.00
F12 :	10.00	F12 :	5.00	F12 :	5.00
F13 :	5.00	F13 :	5.00	F13 :	5.00
F14 :	10.00	F14 :	10.00	F14 :	5.00
F15 :	1.00	F15 :	5.00	F15 :	5.00
F16 :	3.00	F16 :	3.00	F16 :	3.00
F17 :	3.00	F17 :	6.00	F17 :	0.01
A :	0.86	A :	1.22	A :	1.34
B :	1.39	B :	1.35	B :	1.36
C :	0.05	C :	0.01	C :	0.15
Unit_cost :	8.00	Unit_cost :	6.00	Unit_cost :	6.00

Table 6.4: OPTYPE

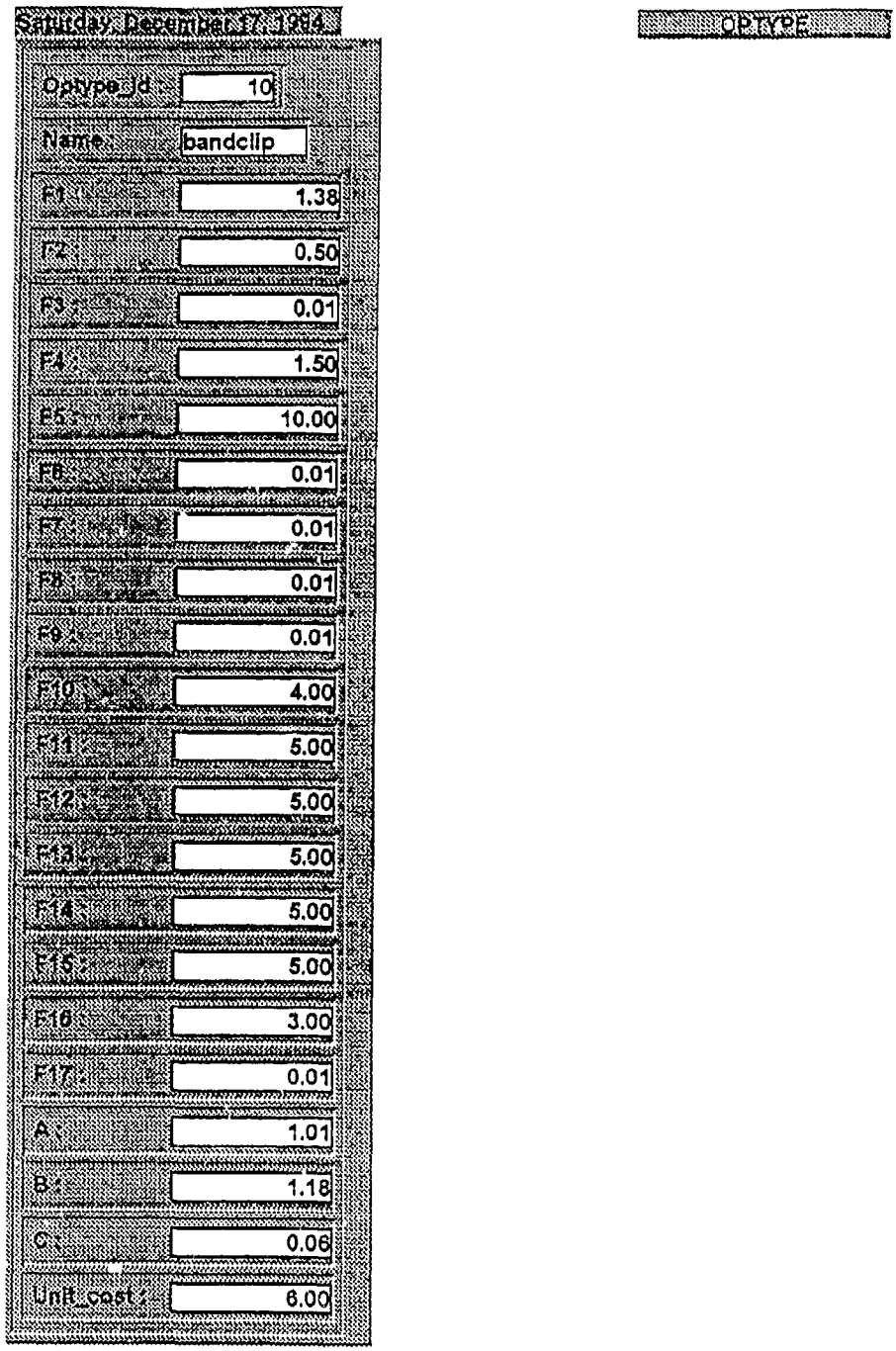


Table 7: STYLE_OP

Saturday, December 17, 1994 STYLE_OP

Style_id	Operation_id	Prev_op_id1	Prev_op_id2	Prev_op_id3	Next_op_id
87	1	-1	0	0	2
87	2	1	0	0	3
87	3	2	0	0	4
87	4	3	0	0	5
87	5	4	0	0	9
87	6	-1	0	0	7
87	7	6	0	0	8
87	8	7	0	0	9
87	9	5	8	0	10
87	10	9	0	0	11
87	11	10	0	0	15
87	12	3	0	0	13
87	13	12	0	0	14
87	14	13	0	0	15
87	15	11	14	0	16
87	16	15	0	0	17
87	17	16	0	0	18
87	18	17	0	0	2
88	2	18	0	0	20
88	8	23	0	0	24
88	13	25	0	0	14
88	14	13	0	0	26
88	19	3	0	0	2
88	20	2	0	0	21
88	21	20	0	0	22
88	22	21	0	0	26
88	23	5	0	0	7
88	24	8	0	0	27
88	25	1	0	0	13
88	26	14	0	0	27
89	27	24	26	0	28
89	28	22	27	0	29
89	29	28	0	0	30
89	30	29	0	0	-2

Table 8 : Master Production Schedule

Jobs	Scheduled Production Quantity							
	1	2	3	4	5	6	7	8
1	1333	579	1111	1111	1111	1111	1112	0
2	965	304	597	597	597	597	597	1097
3	0	523	304	304	304	304	304	304
4	0	188	200	263	273	284	295	458

Jobs	Scheduled Production Quantity							
	9	10	11	12	13	14	15	-
1	0	0	0	0	0	0	0	
2	1175	1167	1217	120	0	0	0	
3	304	305	305	1605	530	0	0	
4	458	458	458	458	1358	1758	91	